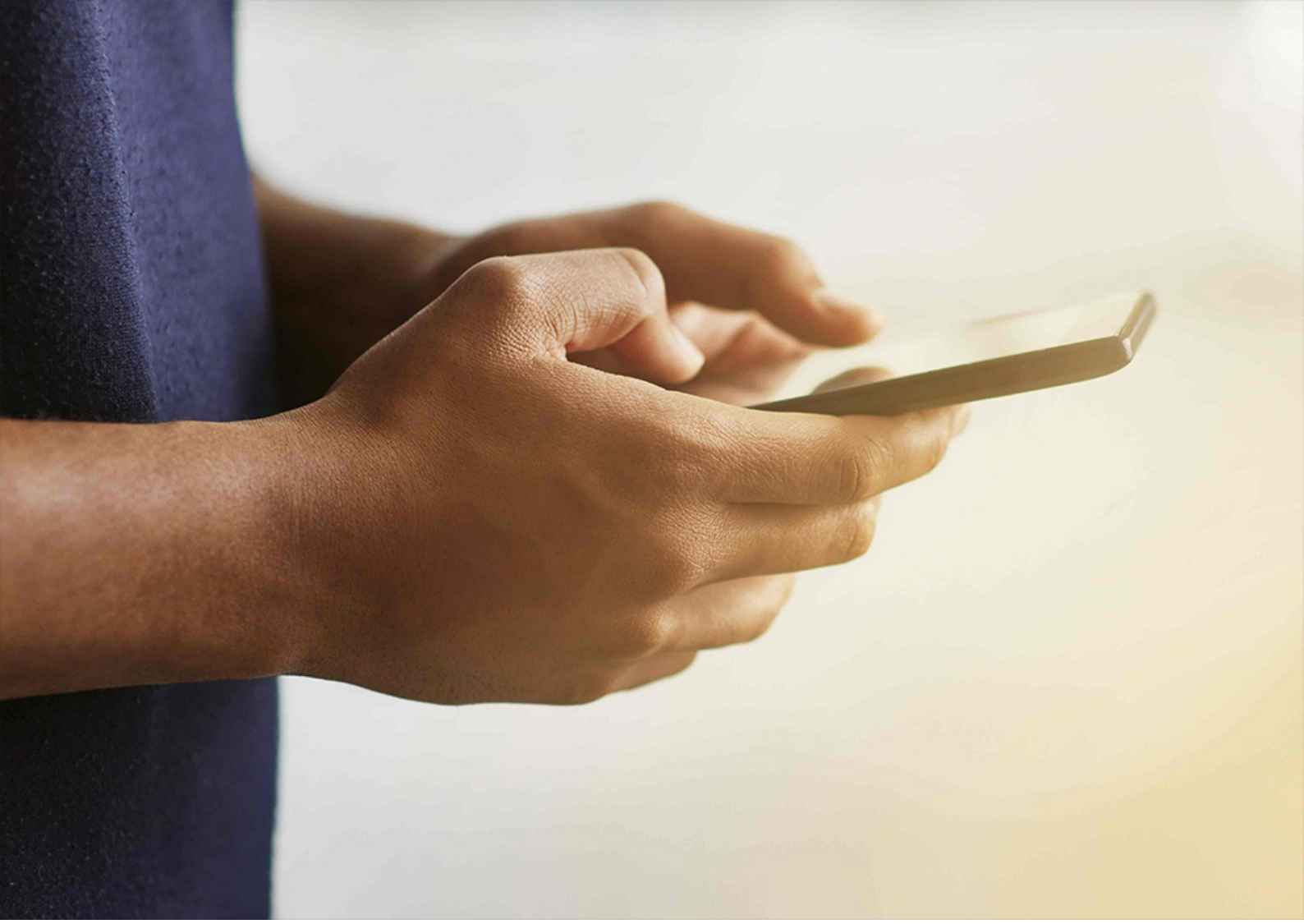


PUBLIC
2026-11-18

Generative AI

© 2026 SAP SE or an SAP affiliate company. All rights reserved.

Content

1 Generative AI Hub 4

1.1 Quick Start 5

Get an Auth Token 6

Get An Orchestration Deployment URL 7

Consume Models with the Harmonized API 8

Next Steps 10

1.2 Regional Availabilty 11

Model Availability by Region 11

Feature Availability by Region 37

Hardware Instance Type Availability by Region 39

1.3 Models 48

Supported Models 50

AWS Bedrock 52

Cohere 57

Mistral 60

NVIDIA 67

OpenAI 69

Perplexity 110

Prior Labs 112

SAP 121

Vertex AI 185

1.4 Model Consumption Methods 190

Foundation Models 190

Batch Consumption 200

Orchestration 210

Tabular AI 603

1.5 Optimizations 652

Prerequisites 652

Evaluations 654

Prompt Optimization 664

Metrics 684

1.6 Additional Concepts 703

Create a Deployment for Orchestration 703

Deployment Lifecycle Management 705

Administration 706

Manage mTLS Certificate Secrets 720

Prompt Registry. 732
Rate Limit Management. 770
Inference Observability. 774

1 Generative AI Hub

The generative AI hub is a capability of SAP AI Core and SAP AI Launchpad that enables access to and orchestration of generative AI models within a governed, enterprise-ready runtime environment.

SAP AI Core provides the runtime and lifecycle management foundation for AI scenarios on SAP Business Technology Platform (SAP BTP). Within this foundation, the generative AI hub allows you to integrate and manage large language models (LLMs) and other generative AI services while leveraging the security, scalability, and operational controls of SAP AI Core.

With the generative AI hub, you can:

- Access supported generative AI models
- Submit prompts and manage model interactions
- Integrate generative AI capabilities into business applications
- Operate generative AI workloads under enterprise governance

The generative AI hub doesn't replace SAP AI Core; it extends it. All model execution, resource management, and operational control are handled by SAP AI Core.

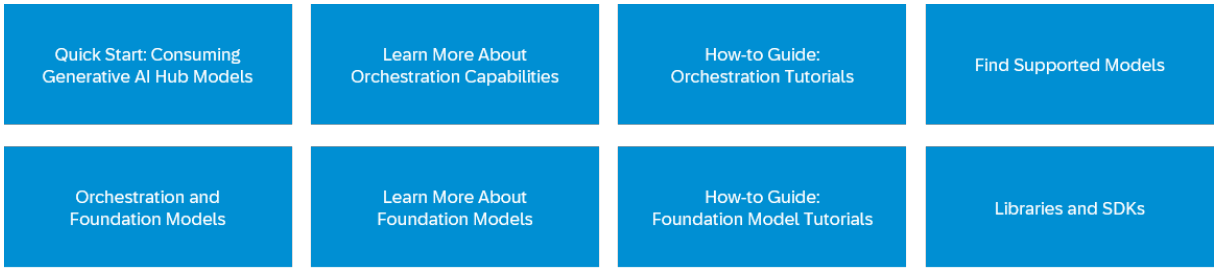
> ⓘ Note
>
> SAP AI Core, including generative AI hub must **not** be used to generate synthetic data for training or fine-tuning models. Any exceptions must be explicitly approved and documented.

For a general overview of the service architecture and runtime capabilities, see What Is SAP AI Core?.

> → Tip
>
> If you want a guided, hands-on environment, you can explore the generative AI hub using the 30-day free trial. For more information, see Try now: 30-Day Basic Trial.

Next Steps

Click the following tiles to find out more about the generative AI hub.



- Quick Start [page 5]
- Orchestration [page 210]

- Orchestration Tutorials [page 603]
- Supported Models [page 50]
- Models [page 48]
- Foundation Models [page 190]
- Foundation Model Tutorials [page 200]
- Libraries and SDKs

Related Information

What Is SAP AI Core?
Initial Setup
Generative AI Hub

1.1 Quick Start

This section walks you through the essential steps for getting started with the generative AI hub in SAP AI Core. You'll authenticate, retrieve your orchestration deployment URL, and make your first model call using the Harmonized API.

> → Tip
>
> If you want a guided, hands-on environment, you can explore the generative AI hub using the 30-day free trial. For more information, see Try now: 30-Day Basic Trial.

Prerequisites

- You've completed the tasks described in Initial Setup.
- You're using the extended service plan, which is required for the generative AI hub.

Procedure

To get started, complete the following steps:

1. Get an Auth Token [page 6]: Obtain an authentication token to securely access SAP AI Core services. You use this token in all subsequent API calls.
2. Get An Orchestration Deployment URL [page 7]: Retrieve the URL of your Orchestration service deployment. This URL is required for making inference requests through the Harmonized API.

3. Consume Models with the Harmonized API [page 8]: Use the Harmonized API to make inference calls without needing to manage individual model deployments. The Harmonized API provides a uniform interface for interacting with supported large language models.
4. Next Steps [page 10]: After completing an initial model call, you can explore grounding, content filtering, data masking, and additional orchestration capabilities.

Related Information

Service Plans

1.1.1 Get an Auth Token

Authentication token retrieval enables secure access to your SAP instance. Set environment variables and use curl commands to obtain and use an access token. Use this process to authenticate API requests in subsequent steps.

Prerequisites

- curl is installed on your operating system.
- You have the following values from your service key:
  - AI API URL
  - Authentication URL
  - Client ID
  - Client secret

Context

Start by setting the required environment variables, which you can get from your SAP AI Core instance.

Procedure

1. Set up your environment as follows:

```
AI_API_URL=<YOUR AI API URL>
AUTH_URL=<YOUR AUTH URL>
CLIENT_ID=<YOUR CLIENT ID>
CLIENT_SECRET=<YOUR CLIENT SECRET>
RESOURCE_GROUP=default
```

2. Obtain the authentication token by sending the following request:

```
TOKEN=`curl --request POST \
  --url "$AUTH_URL/oauth/token" \
  --header "Content-Type: application/x-www-form-urlencoded" \
  --data "grant_type=client_credentials" \
  --data "client_id=$CLIENT_ID" \
  --data "client_secret=$CLIENT_SECRET"'
```

   The response includes an access token.

```
{"access_token":"ey...", "expires_in":7199, "jti":"...",
"token_type":"bearer"}
```

3. Set the token to use it in the following steps:

```
TOKEN=ey...
```

1.1.2 Get An Orchestration Deployment URL

Prerequisites

- You have an SAP AI Core service instance and service key. For more information, see SAP AI Core Initial Setup Documentation.
- You're using the `extended` service plan. For more information, see Service Plans and Update a Service Plan.
- You have completed the client authorization for your preferred user interface. For more information, see Use a Service Key.

Context

Your default resource group contains a running orchestration deployment. You can use this URL across your organization to access orchestration in the generative AI hub.

Procedure

1. Send a GET request to the endpoint `$AI_API_URL/v2/lm/deployments` and include the default resource group in the header.

```
curl --request GET $AI_API_URL/v2/lm/deployments \
    --header "Authorization: Bearer $TOKEN" \
    --header "AI-Resource-Group: default"
```

2. In the response, search for the deployment object where the `configurationName` is `defaultOrchestrationConfig`, and note the `deploymentUrl` associated with this deployment.
3. Set the value of the returned `deploymentUrl` as an environment variable.

```
ORCH_DEPLOYMENT_URL="<deployment_url>"
```

Next Steps

If you cannot locate a RUNNING orchestration deployment, or would like to create additional orchestration deployments, see Create a Deployment for Orchestration [page 703]

1.1.3 Consume Models with the Harmonized API

In this section, we will provide a minimal inference call without any orchestration modules.

A minimal call to orchestration contains only configurations of the required templating and model configuration modules. The curl command below shows how to make such a request.

Sample Code

```
curl --request POST $ORCH_DEPLOYMENT_URL/v2/completion \
--header 'content-type: application/json' \
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP" \
--data-raw '{
  "config": {
   "modules": {
     "prompt_templating": {
       "prompt": {
         "template": [
           {
             "role": "user",
             "content": "Reply with `{{?text}}` in {{?language}}"
           }
         ],
         "defaults": {
           "language": "English"
         }
       },
       "model": {
         "name": "<model>",
         "params": {
           "max_completion_tokens": 50,
           "temperature": 0.1,
           "frequency_penalty": 0,
           "presence_penalty": 0
         }
       }
     }
   }
  },
  "placeholder_values": {
   "text": "Orchestration is Working!",
   "language": "German"
  }
```

```
}'
```

This request configures the templating module with a single user message with two parameters: `text` and `language`. The `language` parameter is also configured with English as the default. The LLM module is configured to use gpt-4o-mini in the latest available version and a set of model parameters. The `input_params` field contains the values for the parameters `text` and `language`. These values are used during this request in the prompt sent to the model.

The response contains a `request_id`, the module results from each module that was executed, and the `orchestration_result`, which includes the response of the call to the model. For more information about modules available for consumption using the Harmoized API, see Orchestration [page 210].

Output Code


```
{
  "request_id": "d8efd339-8ae1-9810-a076-22360875a2fe",
  "intermediate_results": {
    "templating": [
      {
        "content": "Reply with `Orchestration is Working!` in German",
        "role": "user"
      }
    ],
    "llm": {
      "id": "chatcmpl-D2VKMLhpxnszJKVXeNP0k6OcJTQDK",
      "object": "chat.completion",
      "created": 1769489450,
      "model": "gpt-4o-2024-08-06",
      "system_fingerprint": "fp_4a331a0222",
      "choices": [
        {
          "index": 0,
          "message": {
            "role": "assistant",
            "content": "Die Orchestrierung funktioniert!"
          },
          "finish_reason": "stop"
        }
      ],
      "usage": {
        "completion_tokens": 7,
        "prompt_tokens": 19,
        "total_tokens": 26,
        "prompt_tokens_details": {
          "audio_tokens": 0,
          "cached_tokens": 0
        },
        "completion_tokens_details": {
          "accepted_prediction_tokens": 0,
          "audio_tokens": 0,
          "reasoning_tokens": 0,
          "rejected_prediction_tokens": 0
        }
      }
    }
  },
  "final_result": {
    "id": "chatcmpl-D2VKMLhpxnszJKVXeNP0k6OcJTQDK",
    "object": "chat.completion",
    "created": 1769489450,
    "model": "<model><modelversion>",
    "system_fingerprint": "fp_4a331a0222",
    "choices": [
      {
```

```
        "index": 0,
        "message": {
          "role": "assistant",
          "content": "Die Orchestrierung funktioniert!"
        },
        "finish_reason": "stop"
      }
    ],
    "usage": {
      "completion_tokens": 7,
      "prompt_tokens": 19,
      "total_tokens": 26,
      "prompt_tokens_details": {
        "audio_tokens": 0,
        "cached_tokens": 0
      },
      "completion_tokens_details": {
        "accepted_prediction_tokens": 0,
        "audio_tokens": 0,
        "reasoning_tokens": 0,
        "rejected_prediction_tokens": 0
      }
    }
  }
}
```


The templating module result contains the user message with the filled in parameters. The LLM module result contains the response of the model execution. In this example, the LLM module result and the orchestration result are the same. However, they might differ, such as when the output filtering module filters the response.

Related Information

Consumption of GenAI Models Using Orchestration – A Beginner's Guide
Libraries and SDKs

1.1.4 Next Steps

In the section above, we've outlined the basic steps for model consumption.

If you're interested in exploring advanced use cases for enhancing LLM responses with capabilities like grounding, data masking, and content filtering, see Orchestration [page 210].

Alternatively, you can deploy and consume models using direct model access. For more information, see Foundation Models [page 190].

1.2 Regional Availabilty

1.2.1 Model Availability by Region

Use this regional availability matrix for AI models to identify which models and versions are available in your specific geographic region.

How to Read These Tables

In the tables below, models are listed as follows:

Yes	The model is available and accessed from within that region.
Yes*	The model is available but may be accessed from a cross-regional location. This setup harmonizes the model offering across data centers and allows for load balancing and failovers, which help to mitigate data center outages. Data centers are selected based on model availability and data center proximity.
No	The model isn't available in the specified region.

> **Example**
>
> The model mistralai--mistral-small in region US10 is marked with **Yes***, meaning that it is likely to be accessed within that region. However, data centers outside US10 may also be used for the purposes outlined. The same model in region EU10 is marked with **Yes**, meaning that it is used only with data centers in EU10.

For more information about models—including information about **pricing**, **retirement**, and **rate limits**—see SAP Note 3437766. We recommend that you mark this SAP Note as a favorite so that you receive updates whenever it changes.

Access Types

Models on the generative AI hub can be accessed via remote providers such as AWS Bedrock, or hosted by SAP.

- **SAP-hosted:** Models are hosted on SAP's own infrastructure and managed by SAP AI Core.
- **SAP-managed:** Models are hosted on SAP's tenant-isolated Hyperscaler infrastructure and managed by SAP AI Core.
- **Remote:** Models (such as those provided by AWS) are hosted and managed by providers, and are accessed via SAP AI Core.

Model Versions

You have the following options when working with a model:

- You choose to always have the "latest" version of the model. The latest model version can be found in the table. When a version is deprecated, you are automatically moved to the latest version of the model. Be aware that model behavior can change from version to version.

- You choose to work with a specific version of a model. In this case, you must check when the version will be deprecated and take appropriate steps.

For more information, see Create a Deployment [page 194].

Asia — Pacific — Japan (APJ)

The information provided in these tables is maintained on a best-effort basis. For the authoritative list of models available in your landscape, check the model discovery endpoint `$AI_API_URL/v2/lm/scenarios/foundation-models/models`. For more information, see Supported Models [page 50].

> **Note**
>
> Choose the *Columns* dropdown to show or hide additional information about the models.

Executable ID (Access Type)	Model	Version	AP10 Australia (Sydney)	AP11 Singapore	AP12 Singapore	AP20 Australia (Sydney)	AP21 Australia, Southeast (Melbourne)	AP30 Australia (Sydney)	IN30 India (Mumbai)	JP10 Japan (Tokyo)	JP31 Japan (Tokyo/Osaka)	CN40 China (Shanghai)
aws-bedrock (AWS)	amazon--nova-lite	1	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	No
aws-bedrock (AWS)	amazon--nova-lite	2 (latest)	Yes	Yes	No	Yes	Yes	Yes	No	No	No	No
aws-bedrock (AWS)	amazon--nova-micro	1 (latest)	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	No
aws-bedrock (AWS)	amazon--nova-premier	1 (latest)	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	Yes*	No
aws-bedrock (AWS)	amazon--nova-pro	1 (latest)	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	No

Executable ID (Access Type)	Model	Version	AP10 Australia (Sydney)	AP11 Singapore	AP12 Singapore	AP20 Australia (Sydney)	AP21 Australia, Southeast (Melbourne)	AP30 Australia (Sydney)	IN30 India (Mumbai)	JP10 Japan (Tokyo)	JP31 Japan (Tokyo/Osaka)	CN40 China (Shanghai)
aws-bedrock (AWS)	amazon--titan-embed-image	1 (latest)	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes*	No
aws-bedrock (AWS)	amazon--titan-embed-text	1.2	Yes*	No	No	No	No	No	No	Yes*	Yes*	No
aws-bedrock (AWS)	amazon--titan-embed-text	2 (latest)	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes*	No
aws-bedrock (AWS)	anthropic--claude-4.5-haiku	1 (latest)	Yes	Yes	Yes	Yes	Yes*	Yes	Yes	Yes	Yes	No
aws-bedrock (AWS)	anthropic--claude-4.5-opus	1 (latest)	No	No	No	No	No	No	No	No	No	No
aws-bedrock (AWS)	anthropic--claude-4.5-sonnet	1 (latest)	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No
aws-bedrock (AWS)	anthropic--claude-4.6-opus	1	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes*	No
aws-bedrock (AWS)	anthropic--claude-4.6-sonnet	1	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	No

Executable ID (Access Type)	Model	Version	AP10 Australia (Sydney)	AP11 Singapore	AP12 Singapore	AP20 Australia (Sydney)	AP21 Australia, Southeast (Melbourne)	AP30 Australia (Sydney)	IN30 India (Mumbai)	JP10 Japan (Tokyo)	JP31 Japan (Tokyo/Osaka)	CN40 China (Shanghai)
aws-bedrock (AWS)	anthropic--claude-4.7-opus	1	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	No
aws-bedrock (AWS)	anthropic--claude-4.8-opus	1	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	No
aicore-cohere (SAP-hosted)	cohere--command-a-reasoning	2508 (latest)	Yes*	Yes*	No	Yes*	Yes*	Yes*	Yes*	Yes*	No	No
aicore-cohere (SAP-managed)	cohere-re-ranker	3.5 (latest)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes*	No
aicore-cohere (SAP-managed)	cohere-rerank-pro	4	Yes*	Yes*	No	Yes*	Yes*	Yes*	Yes*	Yes*	No	No
gcp-vertexai (Google)	gemini-2.5-flash	001 (latest)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
gcp-vertexai (Google)	gemini-2.5-flash-image	001	Yes*	Yes*	No	Yes*	Yes*	Yes*	Yes*	Yes*	No	No
gcp-vertexai (Google)	gemini-2.5-flash-lite	001	No	No	No	No	No	No	No	No	No	No

Executable ID (Access Type)	Model	Version	AP10 Australia (Sydney)	AP11 Singapore	AP12 Singapore	AP20 Australia (Sydney)	AP21 Australia, Southeast (Melbourne)	AP30 Australia (Sydney)	IN30 India (Mumbai)	JP10 Japan (Tokyo)	JP31 Japan (Tokyo/Osaka)	CN40 China (Shanghai)
gcp-vertexai (Google)	gemini-2.5-pro	001	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
gcp-vertexai (Google)	gemini-3.1-flash-image	001	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	Yes*	No
gcp-vertexai (Google)	gemini-3.1-flash-lite	001	No	No	No	No	No	No	No	No	Yes	No
gcp-vertexai (Google)	gemini-3.5-flash	001 (latest)	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	Yes	No
gcp-vertexai (Google)	gemini-3.5-flash-lite	001	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	Yes*	No
gcp-vertexai (Google)	gemini-3.8-flash-lite	001	Yes*	Yes*	No	Yes*	Yes*	Yes*	Yes*	Yes*	No	No
gcp-vertexai (Google)	gemini-3-pro-image	001	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	Yes*	No
gcp-vertexai (Google)	gemini-embedding	001	Yes	Yes	No	Yes	Yes	Yes	Yes*	Yes	No	No
gcp-vertexai (Google)	gemini-embedding-2	001	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	Yes*	No

Executable ID (Access Type)	Model	Version	AP10 Australia (Sydney)	AP11 Singapore	AP12 Singapore	AP20 Australia (Sydney)	AP21 Australia, Southeast (Melbourne)	AP30 Australia (Sydney)	IN30 India (Mumbai)	JP10 Japan (Tokyo)	JP31 Japan (Tokyo/Osaka)	CN40 China (Shanghai)
azure-openai (Azure)	gpt-4.1	2025-04-14	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	Yes	No
azure-openai (Azure)	gpt-4.1-mini	2025-04-14	Yes*	Yes*	Yes	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*	No
azure-openai (Azure)	gpt-4.1-nano	2025-04-14	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	Yes*	No
azure-openai (Azure)	gpt-4o	2024-05-13	Yes	Yes*	Yes	Yes	Yes	Yes	Yes*	Yes*	Yes	No
azure-openai (Azure)	gpt-4o	2024-08-06 (latest)	Yes*	Yes*	Yes	Yes*	Yes*	Yes*	Yes*	No	Yes	No
azure-openai (Azure)	gpt-4o	2024-11-20	Yes*	Yes*	Yes	Yes*	Yes*	Yes*	Yes*	Yes	Yes	No
azure-openai (Azure)	gpt-4o-mini	2024-07-18	Yes*	Yes	Yes	Yes	Yes	Yes	Yes	Yes*	Yes	No

Executable ID (Access Type)	Model	Version	AP10 Australia (Sydney)	AP11 Singapore	AP12 Singapore	AP20 Australia (Sydney)	AP21 Australia, Southeast (Melbourne)	AP30 Australia (Sydney)	IN30 India (Mumbai)	JP10 Japan (Tokyo)	JP31 Japan (Tokyo/Osaka)	CN40 China (Shanghai)
azure-openai (Azure)	gpt-5	2025-08-07	Yes*	Yes*	Yes	Yes*	Yes*	Yes*	No	Yes*	Yes*	No
azure-openai (Azure)	gpt-5.1	2025-11-13 (latest)	No	No	No	No	No	No	No	No	No	No
azure-openai (Azure)	gpt-5.2	2025-12-11	No	No	No	No	No	No	No	No	No	No
azure-openai (Azure)	gpt-5.3-codex	2026-02-24	Yes*	Yes*	Yes	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*	No
azure-openai (Azure)	gpt-5.4	2026-03-05	Yes*	Yes*	Yes	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*	No
azure-openai (Azure)	gpt-5.4-nano	2026-03-17	Yes*	Yes*	Yes	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*	No
azure-openai (Azure)	gpt-5.5	2026-04-24	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	No
azure-openai (Azure)	gpt-5.6-luna	2026-07-09	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	Yes*	No

Executable ID (Access Type)	Model	Version	AP10 Australia (Sydney)	AP11 Singapore	AP12 Singapore	AP20 Australia (Sydney)	AP21 Australia, Southeast (Melbourne)	AP30 Australia (Sydney)	IN30 India (Mumbai)	JP10 Japan (Tokyo)	JP31 Japan (Tokyo/Osaka)	CN40 China (Shanghai)
azure-openai (Azure)	gpt-5.6-sol	2026-07-09	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	Yes*	No
azure-openai (Azure)	gpt-5.6-terra	2026-07-09	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	Yes*	No
azure-openai (Azure)	gpt-5-mini	2025-08-07	Yes*	Yes*	Yes	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*	No
azure-openai (Azure)	gpt-5-nano	2025-08-07	Yes*	Yes*	Yes	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*	No
azure-openai (Azure)	gpt-realtime	2025-08-28	Yes*	Yes*	Yes	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*	No
aicore-mistralai (SAP-hosted)	mistralai--mistral-large-instruct	2407 (latest)	Yes*	No	No	No	No	No	No	Yes*	No	No
aicore-mistralai (SAP-hosted)	mistralai--mistral-medium	3.5	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	No
aicore-mistralai (SAP-hosted)	mistralai--mistral-medium-instruct	2505 (latest)	Yes*	Yes*	No	Yes*	Yes*	Yes*	Yes*	Yes*	No	No

Executable ID (Access Type)	Model	Version	AP10 Australia (Sydney)	AP11 Singapore	AP12 Singapore	AP20 Australia (Sydney)	AP21 Australia, Southeast (Melbourne)	AP30 Australia (Sydney)	IN30 India (Mumbai)	JP10 Japan (Tokyo)	JP31 Japan (Tokyo/Osaka)	CN40 China (Shanghai)
aicore-mistralai (SAP-hosted)	mistralai--mistral-small	2603	Yes*	Yes*	No	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*	No
aicore-mistralai (SAP-hosted)	mistralai--mistral-small-instruct	2503 (latest)	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	No	No
aicore-nvidia (SAP-managed)	nvidia--llama-3.2-nv-embedqa-1b	2 (latest)	Yes	Yes	No	Yes	Yes	Yes	No	Yes	No	No
azure-openai (Azure)	o1	2024-12-17	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	Yes*	No
azure-openai (Azure)	o3	2025-04-16	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	Yes*	No
azure-openai (Azure)	o3-mini	2025-01-31	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	Yes*	No
azure-openai (Azure)	o4-mini	2025-04-16	Yes*	Yes*	No	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*	No
aicore-sap (SAP-hosted)	sap-abap-1	1 (latest)	Yes*	Yes*	No	Yes*	Yes*	Yes*	Yes*	Yes*	No	No

Executable ID (Access Type)	Model	Version	AP10 Australia (Sydney)	AP11 Singapore	AP12 Singapore	AP20 Australia (Sydney)	AP21 Australia, Southeast (Melbourne)	AP30 Australia (Sydney)	IN30 India (Mumbai)	JP10 Japan (Tokyo)	JP31 Japan (Tokyo/Osaka)	CN40 China (Shanghai)
aicore-sap (SAP-hosted)	sap-rpt-1.5	1	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	No	No
aicore-sap (SAP-hosted)	sap-rpt-1.5-large	1	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	No	No
aicore-sap (SAP-hosted)	sap-rpt-1.6	1	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	No	No
aicore-sap (SAP-hosted)	sap-rpt-1.6-large	1	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	No	No
aicore-sap (SAP-hosted)	sap-rpt-1-large	1	Yes*	Yes*	No	Yes*	Yes*	Yes*	Yes*	Yes*	No	No
aicore-sap (SAP-hosted)	sap-rpt-1-small	1	Yes*	Yes*	No	Yes*	Yes*	Yes*	Yes*	Yes*	No	No
perplexity-ai (remote)	sonar	perplexity-us	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	No	No
perplexity-ai (remote)	sonar-pro	perplexity-us	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	No	No
prior-labs (SAP-hosted)	tabpfn-3.5-plus	2609	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	No	No

Executable ID (Access Type)	Model	Version	AP10 Australia (Sydney)	AP11 Singapore	AP12 Singapore	AP20 Australia (Sydney)	AP21 Australia, Southeast (Melbourne)	AP30 Australia (Sydney)	IN30 India (Mumbai)	JP10 Japan (Tokyo)	JP31 Japan (Tokyo/Osaka)	CN40 China (Shanghai)
azure-openai (Azure)	text-embedding-3-large	1 (latest)	Yes*	Yes*	Yes	Yes*	Yes*	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	text-embedding-3-small	1 (latest)	Yes	Yes	Yes	Yes	Yes*	Yes	No	Yes	Yes*	No
azure-openai (Azure)	text-embedding-ada-002	2 (latest)	Yes	Yes*	No	Yes*	Yes*	Yes*	Yes*	Yes	Yes	No
azure-openai (Azure)	gpt-6-astra	2026-09-03	No	No	No	No	No	No	No	No	No	No
azure-openai (Azure)	gpt-6-luna	2026-09-22	No	No	No	No	No	No	No	No	No	No
azure-openai (Azure)	gpt-6.1-sol	2026-09-22	Yes*	Yes*	Yes*	Yes*	No	Yes*	No	Yes*	Yes*	No
aicore-sap (SAP-hosted)	sap-rpt-ts	1	No	No	No	No	No	No	No	No	No	No
aicore-sap (SAP-hosted)	sap-rpt-decisions	1	No	No	No	No	No	No	No	No	No	No
aicore-open-source	meta--llama-guard-3-8b	20240707	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	No	No

Executable ID (Access Type)	Model	Version	AP10 Australia (Sydney)	AP11 Singapore	AP12 Singapore	AP20 Australia (Sydney)	AP21 Australia, Southeast (Melbourne)	AP30 Australia (Sydney)	IN30 India (Mumbai)	JP10 Japan (Tokyo)	JP31 Japan (Tokyo/Osaka)	CN40 China (Shanghai)
aicore-open-source	ibm--granite-guardian-8b	4.1	Yes*	Yes*	No	Yes*	Yes*	Yes*	No	Yes*	No	No
alibaba-bailian	qwen3.8-max	latest	No	No	No	No	No	No	No	No	No	Yes
alibaba-bailian	qwen3.6-plus	latest	No	No	No	No	No	No	No	No	No	Yes
alibaba-bailian	qwen-3.7-max	latest	No	No	No	No	No	No	No	No	No	Yes
alibaba-bailian	qwen3.7-plus	latest	No	No	No	No	No	No	No	No	No	Yes
alibaba-bailian	qwen3.6-flash	latest	No	No	No	No	No	No	No	No	No	Yes
alibaba-bailian	text-embedding-v4	latest	No	No	No	No	No	No	No	No	No	Yes

Europe — Middle East — Africa (EMEA)

The information provided in these tables is maintained on a best-effort basis. For the authoritative list of models available in your landscape, check the model discovery endpoint `$AI_API_URL/v2/lm/scenarios/foundation-models/models`. For more information, see Supported Models [page 50].

> **Note**
>
> Choose the *Columns* dropdown to show or hide additional information about the models.

Executable ID (Access Type)	Model	Version	AE01 UAE (Dubai)	EU01 Europe (Rot)	EU02 Europe (Frankfurt)	EU10 Europe (Frankfurt)	EU11 Europe (Frankfurt)	EU20 Europe (Netherlands)	EU30 Europe (Frankfurt)	SA30 Saudi Arabia (Dammam)
aws-bedrock (AWS)	amazon--nova-lite	1	No	No	No	Yes	Yes	Yes	Yes	No
aws-bedrock (AWS)	amazon--nova-lite	2 (latest)	No	No	No	Yes	Yes	Yes	Yes	No
aws-bedrock (AWS)	amazon--nova-micro	1 (latest)	No	No	No	Yes	Yes	Yes	Yes	No
aws-bedrock (AWS)	amazon--nova-premier	1 (latest)	No	No	No	No	No	No	No	No
aws-bedrock (AWS)	amazon--nova-pro	1 (latest)	No	No	No	Yes	Yes	Yes	Yes	No
aws-bedrock (AWS)	amazon--titan-embed-image	1 (latest)	No	No	No	Yes	Yes	Yes	Yes	No
aws-bedrock (AWS)	amazon--titan-embed-text	1.2	No	No	No	Yes*	Yes	Yes*	Yes*	No
aws-bedrock (AWS)	amazon--titan-embed-text	2 (latest)	No	No	No	Yes	Yes	Yes	Yes	No
aws-bedrock (AWS)	anthropic--claude-4.5-haiku	1 (latest)	No	No	No	Yes	Yes	Yes	Yes	No
aws-bedrock (AWS)	anthropic--claude-4.5-opus	1 (latest)	No	No	No	Yes	Yes	Yes	Yes	No
aws-bedrock (AWS)	anthropic--claude-4.5-sonnet	1 (latest)	No	No	No	Yes	Yes	Yes	Yes	No
aws-bedrock (AWS)	anthropic--claude-4.6-opus	1	No	No	No	Yes	Yes	Yes	Yes	No

Executable ID (Access Type)	Model	Version	AE01 UAE (Dubai)	EU01 Europe (Rot)	EU02 Europe (Frankfurt)	EU10 Europe (Frankfurt)	EU11 Europe (Frankfurt)	EU20 Europe (Netherlands)	EU30 Europe (Frankfurt)	SA30 Saudi Arabia (Dammam)
aws-bedrock (AWS)	anthropic--claude-4.6-sonnet	1	No	No	No	Yes	Yes	Yes	Yes	No
aws-bedrock (AWS)	anthropic--claude-4.7-opus	1	No	No	No	Yes	Yes	Yes	Yes	No
aws-bedrock (AWS)	anthropic--claude-4.8-opus	1	No	No	No	Yes	Yes	Yes	Yes	No
aicore-cohere (SAP-hosted)	cohere--command-a-reasoning	2508 (latest)	No	Yes	No	Yes	Yes	Yes	Yes	No
aicore-cohere (SAP-managed)	cohere-reranker	3.5 (latest)	No	No	Yes	Yes	Yes	Yes	Yes	Yes
aicore-cohere (SAP-managed)	cohere-rerank-pro	4	No	No	No	Yes	Yes	Yes	Yes	Yes
gcp-vertexai (Google)	gemini-2.5-flash	001 (latest)	No	No	No	Yes	Yes	Yes	Yes	No
gcp-vertexai (Google)	gemini-2.5-flash-image	001	No	No	No	Yes	Yes	Yes	Yes	No
gcp-vertexai (Google)	gemini-2.5-flash-lite	001	No	No	No	Yes	Yes	Yes	Yes	No
gcp-vertexai (Google)	gemini-2.5-pro	001	No	No	No	Yes	Yes	Yes	Yes	No
gcp-vertexai (Google)	gemini-3.1-flash-image	001	No	No	No	Yes	Yes	Yes	Yes	No

Executable ID (Access Type)	Model	Version	AE01 UAE (Dubai)	EU01 Europe (Rot)	EU02 Europe (Frankfurt)	EU10 Europe (Frankfurt)	EU11 Europe (Frankfurt)	EU20 Europe (Netherlands)	EU30 Europe (Frankfurt)	SA30 Saudi Arabia (Dammam)
gcp-vertexai (Google)	gemini-3.1-flash-lite	001	No	No	No	Yes	Yes	Yes	Yes	No
gcp-vertexai (Google)	gemini-3.5-flash	001	No	No	No	Yes	Yes	Yes	Yes	No
gcp-vertexai (Google)	gemini-3.8-flash	001	No	No	No	Yes	Yes	Yes	Yes	No
gcp-vertexai (Google)	gemini-3.5-flash-lite	001	No	No	No	Yes	Yes	Yes	Yes	No
gcp-vertexai (Google)	gemini-3-pro-image	001	No	No	No	No	No	No	No	No
gcp-vertexai (Google)	gemini-embedding	001	No	No	No	Yes	Yes	Yes	Yes	Yes
gcp-vertexai (Google)	gemini-embedding-2	001	No	No	No	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	gpt-4.1	2025-04-14	Yes	No	No	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	gpt-4.1-mini	2025-04-14	No	No	No	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	gpt-4.1-nano	2025-04-14	No	No	No	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	gpt-4o	2024-05-13	No	No	No	Yes	Yes	Yes	Yes	Yes*

Executable ID (Access Type)	Model	Version	AE01 UAE (Dubai)	EU01 Europe (Rot)	EU02 Europe (Frankfurt)	EU10 Europe (Frankfurt)	EU11 Europe (Frankfurt)	EU20 Europe (Netherlands)	EU30 Europe (Frankfurt)	SA30 Saudi Arabia (Dammam)
azure-openai (Azure)	gpt-4o	2024-08-06 (latest)	Yes	No	No	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	gpt-4o	2024-11-20	Yes	Yes	No	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	gpt-4o-mini	2024-07-18	No	No	No	Yes	Yes	Yes	Yes	Yes*
azure-openai (Azure)	gpt-5	2025-08-07	No	No	No	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	gpt-5.1	2025-11-13 (latest)	Yes	No	No	No	No	No	No	No
azure-openai (Azure)	gpt-5.2	2025-12-11	No	No	No	No	No	No	No	No
azure-openai (Azure)	gpt-5.3-codex	2026-02-24	No	No	No	No	No	No	No	No
azure-openai (Azure)	gpt-5.4	2026-03-05	No	No	No	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	gpt-5.4-nano	2026-03-17	No	No	No	No	No	No	No	No
azure-openai (Azure)	gpt-5.5	2026-04-24	No	No	No	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	gpt-5.6-luna	2026-07-09	No	No	No	Yes	Yes	Yes	Yes	No

Executable ID (Access Type)	Model	Version	AE01 UAE (Dubai)	EU01 Europe (Rot)	EU02 Europe (Frankfurt)	EU10 Europe (Frankfurt)	EU11 Europe (Frankfurt)	EU20 Europe (Netherlands)	EU30 Europe (Frankfurt)	SA30 Saudi Arabia (Dammam)
azure-openai (Azure)	gpt-5.6-sol	2026-07-09	No	No	No	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	gpt-5.6-terra	2026-07-09	No	No	No	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	gpt-5-mini	2025-08-07	Yes	No	No	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	gpt-5-nano	2025-08-07	No	No	No	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	gpt-realtime	2025-08-28	No	No	No	Yes*	No	Yes*	Yes*	No
aicore-mistralai (SAP-hosted)	mistralai--mistral-large-instruct	2407 (latest)	No	Yes	No	Yes	Yes	Yes	Yes	No
aicore-mistralai (SAP-hosted)	mistralai--mistral-medium	3.5	No	Yes	Yes	Yes	Yes	Yes	Yes	No
aicore-mistralai (SAP-hosted)	mistralai--mistral-medium-instruct	2505 (latest)	No	Yes	No	Yes	Yes	Yes	Yes	No
aicore-mistralai (SAP-hosted)	mistralai--mistral-small	2603	No	Yes	No	Yes	Yes	Yes	Yes	No
aicore-mistralai (SAP-hosted)	mistralai--mistral-small-instruct	2503 (latest)	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Executable ID (Access Type)	Model	Version	AE01 UAE (Dubai)	EU01 Europe (Rot)	EU02 Europe (Frankfurt)	EU10 Europe (Frankfurt)	EU11 Europe (Frankfurt)	EU20 Europe (Netherlands)	EU30 Europe (Frankfurt)	SA30 Saudi Arabia (Dammam)
aicore-nvidia (SAP-managed)	nvidia--llama-3.2-nv-embedqa-1b	2 (latest)	No	Yes	Yes*	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	o1	2024-12-17	No	No	No	Yes	Yes	No	No	No
azure-openai (Azure)	o3	2025-04-16	No	No	No	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	o3-mini	2025-01-31	Yes	No	No	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	o4-mini	2025-04-16	No	No	No	Yes	Yes	Yes	Yes	No
aicore-sap (SAP-hosted)	sap-abap-1	1 (latest)	No	Yes	No	Yes	Yes	Yes	Yes	No
aicore-sap (SAP-hosted)	sap-rpt-1.5	1	No	Yes	No	Yes	Yes	Yes	Yes	No
aicore-sap (SAP-hosted)	sap-rpt-1.5-large	1	No	Yes	No	Yes	Yes	Yes	Yes	No
aicore-sap (SAP-hosted)	sap-rpt-1.6	1	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
aicore-sap (SAP-hosted)	sap-rpt-1.6-large	1	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
aicore-sap (SAP-hosted)	sap-rpt-1-large	1	No	Yes	No	Yes	Yes	Yes	Yes	No
aicore-sap (SAP-hosted)	sap-rpt-1-small	1	No	Yes	No	Yes	Yes	Yes	Yes	No

Executable ID (Access Type)	Model	Version	AE01 UAE (Dubai)	EU01 Europe (Rot)	EU02 Europe (Frankfurt)	EU10 Europe (Frankfurt)	EU11 Europe (Frankfurt)	EU20 Europe (Netherlands)	EU30 Europe (Frankfurt)	SA30 Saudi Arabia (Dammam)
perplexity-ai (remote)	sonar	perplexity-us	No	No	No	Yes*	No	Yes*	Yes*	No
perplexity-ai (remote)	sonar-pro	perplexity-us	No	No	No	Yes*	No	Yes*	Yes*	No
prior-labs (SAP-hosted)	tabpfn-3.5-plus	2609	No	Yes	No	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	text-embedding-3-large	1 (latest)	Yes	Yes	No	Yes*	Yes	Yes*	Yes*	Yes*
azure-openai (Azure)	text-embedding-3-small	1 (latest)	Yes	No	No	Yes*	Yes	Yes*	Yes*	No
azure-openai (Azure)	text-embedding-ada-002	2 (latest)	Yes	No	No	Yes	Yes	Yes	Yes	Yes*
azure-openai (Azure)	gpt-6-astra	2026-09-03	No	No	No	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	gpt-6-luna	2026-09-22	No	No	No	Yes	Yes	Yes	Yes	No
azure-openai (Azure)	gpt-6.1-sol	2026-09-22	No	No	No	Yes	Yes	Yes	Yes	No
aicore-sap (SAP-hosted)	sap-rpt-ts	1	No	Yes	No	Yes	Yes	Yes	Yes	No
aicore-sap (SAP-hosted)	sap-rpt-decisions	1	No	No	No	Yes	No	No	No	No

Executable ID (Access Type)	Model	Version	AE01 UAE (Dubai)	EU01 Europe (Rot)	EU02 Europe (Frankfurt)	EU10 Europe (Frankfurt)	EU11 Europe (Frankfurt)	EU20 Europe (Netherlands)	EU30 Europe (Frankfurt)	SA30 Saudi Arabia (Dammam)
aicore-open-source	meta--llama-guard-3-8b	202407	No	No	No	Yes*	No	Yes*	Yes*	No
aicore-open-source	ibm--granite-guardian-8b	4.1	No	No	No	Yes	No	Yes	Yes	No
alibaba-bailian	qwen3.8-max	latest	No	No	No	No	No	No	No	Yes
alibaba-bailian	qwen-3.7-max	latest	No	No	No	No	No	No	No	Yes
alibaba-bailian	qwen3.7-plus	latest	No	No	No	No	No	No	No	Yes
alibaba-bailian	qwen3.6-flash	latest	No	No	No	No	No	No	No	Yes
alibaba-bailian	text-embedding-v4	latest	No	No	No	No	No	No	No	Yes

Americas (AMER)

The information provided in these tables is maintained on a best-effort basis. For the authoritative list of models available in your landscape, check the model discovery endpoint `$AI_API_URL/v2/lm/scenarios/foundation-models/models`. For more information, see Supported Models [page 50].

> **Note**
>
> Choose the *Columns* dropdown to show or hide additional information about the models.

Executable ID (Access Type)	Model	Version	BR10 Brazil (São Paulo)	CA10 Canada (Montréal)	US10 US East (N. Virginia)	US20 US West (Oregon)	US21 US East (Virginia)	US30 US Central (Iowa)
aws-bedrock (AWS)	amazon--nova-lite	1	No	No	Yes	Yes	Yes	Yes

Executable ID (Access Type)	Model	Version	BR10 Brazil (São Paulo)	CA10 Canada (Montréal)	US10 US East (N. Virginia)	US20 US West (Oregon)	US21 US East (Virginia)	US30 US Central (Iowa)
aws-bedrock (AWS)	amazon--nova-lite	2 (latest)	Yes*	No	Yes	Yes	Yes	Yes
aws-bedrock (AWS)	amazon--nova-micro	1 (latest)	No	No	Yes	Yes	Yes	Yes
aws-bedrock (AWS)	amazon--nova-premier	1 (latest)	No	Yes	Yes	Yes	Yes	Yes
aws-bedrock (AWS)	amazon--nova-pro	1 (latest)	No	No	Yes	Yes	Yes	Yes
aws-bedrock (AWS)	amazon--titan-embed-image	1 (latest)	Yes	No	Yes	Yes	Yes	Yes
aws-bedrock (AWS)	amazon--titan-embed-text	1.2	No	No	Yes	Yes	Yes	Yes
aws-bedrock (AWS)	amazon--titan-embed-text	2 (latest)	No	No	Yes	Yes	Yes	Yes
aws-bedrock (AWS)	anthropic--claude-4.5-haiku	1 (latest)	Yes	No	Yes	Yes	Yes	Yes
aws-bedrock (AWS)	anthropic--claude-4.5-opus	1 (latest)	No	No	Yes	Yes	Yes	Yes
aws-bedrock (AWS)	anthropic--claude-4.5-sonnet	1 (latest)	Yes	No	Yes	Yes	Yes	Yes
aws-bedrock (AWS)	anthropic--claude-4.6-opus	1	Yes	No	Yes	Yes	Yes	Yes
aws-bedrock (AWS)	anthropic--claude-4.6-sonnet	1	Yes	No	Yes	Yes	Yes	Yes
aws-bedrock (AWS)	anthropic--claude-4.7-opus	1	Yes	No	Yes	Yes	Yes	Yes
aws-bedrock (AWS)	anthropic--claude-4.8-opus	1	No	No	Yes	Yes	Yes	Yes

Executable ID (Access Type)	Model	Version	BR10 Brazil (São Paulo)	CA10 Canada (Montréal)	US10 US East (N. Virginia)	US20 US West (Oregon)	US21 US East (Virginia)	US30 US Central (Iowa)
aicore-cohere (SAP-hosted)	cohere--command-a-reasoning	2508 (latest)	Yes*	No	Yes*	Yes	Yes*	Yes*
aicore-cohere (SAP-managed)	cohere-re-ranker	3.5 (latest)	Yes	Yes	Yes	Yes	Yes	Yes
aicore-cohere (SAP-managed)	cohere-re-rank-pro	4	No	No	Yes*	Yes*	Yes*	Yes*
gcp-vertexai (Google)	gemini-2.5-flash	001 (latest)	Yes	Yes	Yes	Yes	Yes	Yes
gcp-vertexai (Google)	gemini-2.5-flash-image	001	Yes*	No	Yes	Yes	Yes	Yes
gcp-vertexai (Google)	gemini-2.5-flash-lite	001	No	No	Yes	Yes	Yes	Yes
gcp-vertexai (Google)	gemini-2.5-pro	001	Yes	Yes	Yes	Yes	Yes	Yes
gcp-vertexai (Google)	gemini-3.1-flash-image	001	No	No	Yes*	Yes	Yes*	Yes*
gcp-vertexai (Google)	gemini-3.1-flash-lite	001	No	No	Yes	Yes	Yes	Yes
gcp-vertexai (Google)	gemini-3.5-flash	001 (latest)	Yes*	No	Yes	Yes	Yes	Yes
gcp-vertexai (Google)	gemini-3.8-flash	001	Yes*	Yes*	Yes	Yes*	Yes*	Yes*
gcp-vertexai (Google)	gemini-3.5-flash-lite	001	Yes*	No	Yes	Yes	Yes	Yes
gcp-vertexai (Google)	gemini-3-pro-image	001	No	No	Yes*	Yes	Yes*	Yes*
gcp-vertexai (Google)	gemini-embedding	001	Yes*	No	Yes	Yes	Yes	Yes
gcp-vertexai (Google)	gemini-embedding-2	001	No	No	Yes	No	Yes	Yes
azure-openai (Azure)	gpt-4.1	2025-04-14	Yes*	No	Yes	Yes	Yes	Yes
azure-openai (Azure)	gpt-4.1-mini	2025-04-14	Yes*	Yes	Yes	Yes	Yes	Yes

Executable ID (Access Type)	Model	Version	BR10 Brazil (São Paulo)	CA10 Canada (Montréal)	US10 US East (N. Virginia)	US20 US West (Oregon)	US21 US East (Virginia)	US30 US Central (Iowa)
azure-openai (Azure)	gpt-4.1-nano	2025-04-14	No	No	Yes	Yes	Yes	Yes
azure-openai (Azure)	gpt-4o	2024-05-13	Yes*	Yes	Yes	Yes	Yes	Yes
azure-openai (Azure)	gpt-4o	2024-08-06 (latest)	Yes*	No	Yes	Yes	Yes	Yes
azure-openai (Azure)	gpt-4o	2024-11-20	Yes*	No	Yes	Yes	Yes	Yes
azure-openai (Azure)	gpt-4o-mini	2024-07-18	Yes*	No	Yes	Yes	Yes	Yes
azure-openai (Azure)	gpt-5	2025-08-07	Yes*	No	Yes	Yes	Yes	Yes
azure-openai (Azure)	gpt-5.1	2025-11-13 (latest)	No	No	No	Yes	No	No
azure-openai (Azure)	gpt-5.2	2025-12-11	No	No	Yes	Yes	Yes	Yes
azure-openai (Azure)	gpt-5.3-codex	2026-02-24	Yes*	No	Yes	Yes	Yes	Yes
azure-openai (Azure)	gpt-5.4	2026-03-05	Yes*	No	Yes	Yes	Yes	Yes
azure-openai (Azure)	gpt-5.4-nano	2026-03-17	Yes*	No	Yes	Yes	Yes	Yes
azure-openai (Azure)	gpt-5.5	2026-04-24	Yes	No	Yes	Yes	Yes	Yes
azure-openai (Azure)	gpt-5.6-luna	2026-07-09	Yes*	No	Yes	No	Yes	Yes

Executable ID (Access Type)	Model	Version	BR10 Brazil (São Paulo)	CA10 Canada (Montréal)	US10 US East (N. Virginia)	US20 US West (Oregon)	US21 US East (Virginia)	US30 US Central (Iowa)
azure-openai (Azure)	gpt-5.6-sol	2026-07-09	Yes*	No	Yes	No	Yes	Yes
azure-openai (Azure)	gpt-5.6-terra	2026-07-09	Yes*	No	Yes	No	Yes	Yes
azure-openai (Azure)	gpt-5-mini	2025-08-07	Yes*	No	Yes	Yes	Yes	Yes
azure-openai (Azure)	gpt-5-nano	2025-08-07	Yes*	No	Yes	Yes	Yes	Yes
azure-openai (Azure)	gpt-realtime	2025-08-28	No	No	Yes*	Yes	Yes*	Yes*
aicore-mistralai (SAP-hosted)	mistralai--mistral-large-instruct	2407 (latest)	No	No	Yes*	Yes	Yes*	Yes*
aicore-mistralai (SAP-hosted)	mistralai--mistral-medium	3.5	Yes	No	Yes	Yes	Yes	Yes
aicore-mistralai (SAP-hosted)	mistralai--mistral-medium-instruct	2505 (latest)	Yes*	No	Yes*	Yes	Yes*	Yes*
aicore-mistralai (SAP-hosted)	mistralai--mistral-small	2603	No	No	Yes*	Yes	Yes*	Yes*
aicore-mistralai (SAP-hosted)	mistralai--mistral-small-instruct	2503 (latest)	No	No	Yes*	Yes	Yes*	Yes*
aicore-nvidia (SAP-managed)	nvidia--llama-3.2-nv-embedqa-1b	2 (latest)	No	No	Yes	Yes	Yes	Yes
azure-openai (Azure)	o1	2024-12-17	No	No	Yes*	Yes	Yes*	Yes*
azure-openai (Azure)	o3	2025-04-16	No	No	Yes*	Yes	Yes*	Yes*

Executable ID (Access Type)	Model	Version	BR10 Brazil (São Paulo)	CA10 Canada (Montréal)	US10 US East (N. Virginia)	US20 US West (Oregon)	US21 US East (Virginia)	US30 US Central (Iowa)
azure-openai (Azure)	o3-mini	2025-01-31	No	No	Yes	Yes	Yes	Yes
azure-openai (Azure)	o4-mini	2025-04-16	Yes*	No	Yes*	Yes	Yes*	Yes*
aicore-sap (SAP-hosted)	sap-abap-1	1 (latest)	Yes*	No	Yes*	Yes	Yes*	Yes*
aicore-sap (SAP-hosted)	sap-rpt-1.5	1	Yes	No	Yes	Yes	Yes	Yes
aicore-sap (SAP-hosted)	sap-rpt-1.5-large	1	Yes	No	Yes	Yes	Yes	Yes
aicore-sap (SAP-hosted)	sap-rpt-1.6	1	Yes*	No	Yes*	Yes*	Yes*	Yes*
aicore-sap (SAP-hosted)	sap-rpt-1.6-large	1	Yes*	No	Yes*	Yes*	Yes*	Yes*
aicore-sap (SAP-hosted)	sap-rpt-1-large	1	Yes*	No	Yes*	Yes	Yes*	Yes*
aicore-sap (SAP-hosted)	sap-rpt-1-small	1	Yes*	No	Yes*	Yes	Yes*	Yes*
perplexity-ai (remote)	sonar	perplexity-us	Yes*	No	Yes	Yes	Yes	Yes
perplexity-ai (remote)	sonar-pro	perplexity-us	Yes*	No	Yes	Yes	Yes	Yes
prior-labs (SAP-hosted)	tabpfn-3.5-plus	2609	Yes*	No	Yes*	Yes*	Yes*	Yes*
azure-openai (Azure)	text-embedding-3-large	1 (latest)	Yes*	Yes	Yes	Yes	Yes	Yes

Executable ID (Access Type)	Model	Version	BR10 Brazil (São Paulo)	CA10 Canada (Montréal)	US10 US East (N. Virginia)	US20 US West (Oregon)	US21 US East (Virginia)	US30 US Central (Iowa)
azure-openai (Azure)	text-embedding-3-small	1 (latest)	No	Yes	Yes	Yes	Yes	Yes
azure-openai (Azure)	text-embedding-ada-002	2 (latest)	Yes*	Yes	Yes	Yes	Yes*	Yes*
azure-openai (Azure)	gpt-6-astra	2026-09-03	No	No	Yes	Yes	Yes	Yes*
azure-openai (Azure)	gpt-6-luna	2026-09-22	No	No	Yes	Yes	Yes	Yes*
azure-openai (Azure)	gpt-6.1-sol	2026-09-22	No	No	Yes	Yes	Yes	Yes*
aicore-sap (SAP-hosted)	sap-rpt-ts	1	No	No	Yes	Yes	Yes	Yes*
aicore-sap (SAP-hosted)	sap-rpt-decisions	1	No	No	No	No	No	No
aicore-open-source	meta--llama-guard-3-8b	2024 07	No	No	Yes*	No	Yes*	Yes*
aicore-open-source	ibm--granite-guardian-8b	4.1	No	No	Yes*	No	Yes*	Yes*

Compliance

SAP AI Core, including generative AI hub must **not** be used to generate synthetic data for training or fine-tuning models. Any exceptions must be explicitly approved and documented.

You are responsible for making sure that your use of AI models on the generative AI hub complies with applicable laws, regulations, rules, and corporate policies. Note that this statement does not modify SAP's obligations under your agreement for the use of the generative AI hub.

1.2.2 Feature Availability by Region

Use this regional availability matrix to find out which generative AI features are available across SAP's global data centers. The tables show which AI capabilities (such as LLM Orchestration and Prompt Registry) are accessible in your specific geographic region.

How to Read These Tables

In the tables below, AI and machine learning features are listed as follows:

Yes	The feature is available and accessed from within that region.
No	The feature isn't available in the specified region.

Information provided in these tables is maintained on a best-effort basis.

Asia — Pacific — Japan (APJ)

> **Note**
>
> Choose the *Columns* dropdown to show or hide additional information about the features.

Feature	AP10 Australia (Sydney)	AP11 Singapore	AP12 Singapore	AP20 Australia (Sydney)	AP21 Australia, Southeast (Melbourne)	AP30 Australia (Sydney)	IN30 India (Mumbai)	JP10 Japan (Tokyo)	JP31 Japan (Tokyo/Osaka)
Tabular AI	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	No
LLM Batch Service	No	No	No	No	No	No	No	No	No
Context Registry	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	No
LLM Orchestration	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Feedback Service	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
LLM Evaluation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Prompt Optimization	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
Prompt Registry	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Document Grounding	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Europe — Middle East — Africa (EMEA)

> **Note**
>
> Choose the *Columns* dropdown to show or hide additional information about the features.

Feature	AE01 UAE (Dubai)	EU01 Europe (Rot)	EU02 Europe (Frankfurt)	EU10 Europe (Frankfurt)	EU11 Europe (Frankfurt)	EU20 Europe (Netherlands)	EU30 Europe (Frankfurt)	SA30 Saudi Arabia (Dammam)
Tabular AI	No	Yes	No	Yes	No	Yes	Yes	No
LLM Batch Service	No	Yes	No	Yes	No	Yes	Yes	No
Context Registry	No	Yes	No	Yes	Yes	Yes	Yes	No
LLM Orchestration	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Feedback Service	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
LLM Evaluation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Prompt Optimization	No	No	No	Yes	Yes	Yes	Yes	No
Prompt Registry	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Document Grounding	No	Yes	No	Yes	Yes	Yes	Yes	Yes

Americas (AMER)

> **Note**
>
> Choose the *Columns* dropdown to show or hide additional information about the features.

Feature	BR10 Brazil (São Paulo)	CA10 Canada (Montréal)	US10 US East (N. Virginia)	US20 US West (Oregon)	US21 US East (Virginia)	US30 US Central (Iowa)
Tabular AI	Yes	No	Yes	Yes	Yes	Yes
LLM Batch Service	No	No	Yes	Yes	Yes	Yes
Context Registry	Yes	No	Yes	Yes	Yes	Yes
LLM Orchestration	Yes	Yes	Yes	Yes	Yes	Yes
Feedback Service	Yes	Yes	Yes	Yes	Yes	Yes
LLM Evaluation	Yes	Yes	Yes	Yes	Yes	Yes
Prompt Optimization	Yes	Yes	Yes	Yes	Yes	Yes

Feature	BR10 Brazil (São Paulo)	CA10 Canada (Montréal)	US10 US East (N. Virginia)	US20 US West (Oregon)	US21 US East (Virginia)	US30 US Central (Iowa)
Prompt Registry	Yes	Yes	Yes	Yes	Yes	Yes
Document Grounding	Yes	No	Yes	No	Yes	Yes

1.2.3 Hardware Instance Type Availability by Region

Overview

This topic provides an overview of the available instance types and specifications across supported data centers. It also includes information about node-hour conversion rates for each instance type.

Important Information

- You can request new hardware instance types by SNOW ticket. For more information, see Support Process.
- Include the following information in your request:
  - CPU, GPU, memory, and storage requirements
  - Preferred region
  - Business justification
  - Required delivery timeline or urgency
  - Contact information
- Requests are evaluated based on compliance and security requirements, commercial feasibility, regional availability and quota constraints, and delivery capacity. If the requested instance type cannot be provided, an alternative region or instance type may be proposed.
- For guidance on selecting instance types in SAP AI Core, see Choose an Instance Type.
- Resource plans are not supported in newly introduced data centers. Use instance types instead.

Availability and Pricing Information

The information in the following tables is subject to change, including availability, pricing, and general availability status.

Additional information about conversion rates, regional availability, and supported configurations is provided after the tables.

How to Read These Tables

In the tables below, AI and machine learning instance types are listed as follows:

Yes	The instance type is available and accessed from within that region.
No	The instance type isn't available in the specified region.

Information provided in these tables is maintained on a best-effort basis.

Asia — Pacific — Japan (APJ)

Provider	Instance Plan ID	Compute Factor	GPUs	GPU Memory (GB)	CPU Cores	Memory (GB)	AP10 Australia (Sydney)	AP11 Singapore	AP12 Singapore	AP20 Australia (Sydney)	AP21 Australia, Southeast (Melbourne)	AP30 Australia (Sydney)	IN30 India (Mumbai)	JP10 Japan (Tokyo)	JP31 Japan (Tokyo/Osaka)
AWS	m7i.xlarge	0.44536	-	-	4	16	Yes	Yes	Yes	No	No	No	No	Yes	No
AWS	m7i.2xlarge	0.81903	-	-	8	32	Yes	Yes	Yes	No	No	No	No	Yes	No
AWS	m7i.4xlarge	1.56641	-	-	16	64	Yes	Yes	Yes	No	No	No	No	Yes	No
AWS	m7i.8xlarge	3.02526	-	-	32	128	Yes	Yes	Yes	No	No	No	No	Yes	No
AWS	g4dn.xlarge	0.98727	1 T4	16	4	16	Yes	Yes	Yes	No	No	No	No	Yes	No
AWS	g4dn.2xlarge	1.49228	1 T4	16	8	32	Yes	Yes	Yes	No	No	No	No	Yes	No
AWS	g4dn.4xlarge	2.36363	1 T4	16	16	64	Yes	Yes	Yes	No	No	No	No	Yes	No
AWS	g6.4xlarge	2.47566	1 L4	24	16	64	Yes	Yes	Yes	No	No	No	No	Yes	No
AWS	g6.12xlarge	8.07090	4 L4	96	96	384	Yes	Yes	Yes	No	No	No	No	Yes	No
AWS	g6e.4xlarge	5.33927	1 L40S	48	16	128	No	No	Yes	No	No	No	No	Yes	No

Provider	Instance Plan ID	Compute Factor	GPUs	GPU Memory (GB)	CPU Cores	Memory (GB)	AP10 Australia (Sydney)	AP11 Singapore	AP12 Singapore	AP20 Australia (Sydney)	AP21 Australia, Southeast (Melbourne)	AP30 Australia (Sydney)	IN30 India (Mumbai)	JP10 Japan (Tokyo)	JP31 Japan (Tokyo/Osaka)
AWS	p4d.24xlarge	37.97723	8 A100	320	96	1152	No	Yes	No	No	No	No	No	No	No
GCP	n4-standard-4	0.37530	-	-	4	16	No	No	No	No	No	No	Yes	No	Yes
GCP	n4-standard-8	0.78152	-	-	8	32	No	No	No	No	No	No	Yes	No	Yes
GCP	n4-standard-16	1.50125	-	-	16	64	No	No	No	No	No	No	Yes	No	Yes
GCP	n4-standard-32	2.90973	-	-	32	128	No	No	No	No	No	No	Yes	No	Yes
GCP	n1-standard-4	0.36205	1 T4	16	4	15	No	No	No	No	No	Yes	Yes	No	No
GCP	n1-standard-8	0.72411	1 T4	16	8	30	No	No	No	No	No	No	Yes	No	Yes
GCP	n1-standard-16	1.29359	1 T4	16	16	60	No	No	No	No	No	Yes	Yes	No	No
GCP	g2-standard-16	2.13636	1 L4	24	16	64	No	No	No	No	No	No	Yes	No	Yes
GCP	a2-ultragpu-1g	8.03933	1 A100	80	12	170	No	No	No	No	No	No	No	No	Yes
GCP	a3-high-gpu-1g	16.71826	8 H100	640	208	1872	No	No	No	No	No	No	No	No	Yes

Provider	Instance Plan ID	Compute Factor	GPUs	GPU Memory (GB)	CPU Cores	Memory (GB)	AP10 Australia (Sydney)	AP11 Singapore	AP12 Singapore	AP20 Australia (Sydney)	AP21 Australia, Southeast (Melbourne)	AP30 Australia (Sydney)	IN30 India (Mumbai)	JP10 Japan (Tokyo)	JP31 Japan (Tokyo/Osaka)
GCP	n4-standard-4	0.37530	-	-	4	16	No	No	No	No	No	No	Yes	No	Yes
GCP	n4-standard-8	0.78152	-	-	8	32	No	No	No	No	No	No	Yes	No	Yes
GCP	n4-standard-16	1.50125	-	-	16	64	No	No	No	No	No	No	Yes	No	Yes
GCP	n4-standard-32	2.90973	-	-	32	128	No	No	No	No	No	No	Yes	No	Yes
GCP	n1-standard-4	0.36205	1 T4	16	4	15	No	No	No	No	No	Yes	Yes	No	No
GCP	n1-standard-8	0.72411	1 T4	16	8	30	No	No	No	No	No	No	Yes	No	Yes
GCP	n1-standard-16	1.29359	1 T4	16	16	60	No	No	No	No	No	Yes	Yes	No	No
GCP	g2-standard-16	2.13636	1 L4	24	16	64	No	No	No	No	No	No	Yes	No	Yes
GCP	a2-ultragpu-1g	8.03933	1 A100	80	12	170	No	No	No	No	No	No	No	No	Yes
GCP	a3-high-gpu-1g	16.71826	8 H100	640	208	1872	No	No	No	No	No	No	No	No	Yes

Europe — Middle East — Africa (EMEA)

Provider	Instance Plan ID	Compute Factor	GPUs	GPU Memory (GB)	CPU Cores	Memory (GB)	AE01 UAE (Dubai)	EU01 Europe (Rotterdam)	EU02 Europe (Frankfurt)	EU10 Europe (Frankfurt)	EU11 Europe (Frankfurt)	EU20 Europe (Netherlands)	EU30 Europe (Frankfurt)	SA30 Saudi Arabia (Dammam)
AWS	m7i.xlarge	0.44536	-	-	4	16	No	No	No	Yes	Yes	No	No	No
AWS	m7i.2xlarge	0.81903	-	-	8	32	No	No	No	Yes	Yes	No	No	No
AWS	m7i.4xlarge	1.56641	-	-	16	64	No	No	No	Yes	Yes	No	No	No
AWS	m7i.8xlarge	3.02526	-	-	32	128	No	No	No	Yes	Yes	No	No	No
AWS	g4dn.xlarge	0.98727	1 T4	16	4	16	No	No	No	Yes	Yes	No	No	No
AWS	g4dn.2xlarge	1.49228	1 T4	16	8	32	No	No	No	Yes	Yes	No	No	No
AWS	g4dn.4xlarge	2.36363	1 T4	16	16	64	No	No	No	Yes	Yes	No	No	No
AWS	g6.4xlarge	2.47566	1 L4	24	16	64	No	No	No	Yes	Yes	No	No	No
AWS	g6.12xlarge	8.07090	4 L4	96	96	384	No	No	No	Yes	Yes	No	No	No
AWS	g6e.4xlarge	5.33927	1 L40S	48	16	128	No	No	No	Yes	Yes	No	No	No
AWS	p4d.24xlarge	37.97723	8 A100	320	96	1152	No	No	No	No	No	No	No	No
GCP	n4-standard-4	0.37530	-	-	4	16	No	No	No	No	No	No	Yes	Yes
GCP	n4-standard-8	0.78152	-	-	8	32	No	No	No	No	No	No	Yes	Yes
GCP	n4-standard-16	1.50125	-	-	16	64	No	No	No	No	No	No	Yes	Yes

Provider	Instance Plan ID	Compute Factor	GPUs	GPU Memory (GB)	CPU Cores	Memory (GB)	AE01 UAE (Dubai)	EU01 Europe (Rotterdam)	EU02 Europe (Frankfurt)	EU10 Europe (Frankfurt)	EU11 Europe (Frankfurt)	EU20 Europe (Netherlands)	EU30 Europe (Frankfurt)	SA30 Saudi Arabia (Dammam)
GCP	n4-standard-32	2.90973	-	-	32	128	No	No	No	No	No	No	Yes	Yes
GCP	n1-standard-4	0.36205	1 T4	16	4	15	No	No	No	No	No	No	Yes	No
GCP	n1-standard-8	0.72411	1 T4	16	8	30	No	No	No	No	No	No	Yes	No
GCP	n1-standard-16	1.29359	1 T4	16	16	60	No	No	No	No	No	No	Yes	No
GCP	g2-standard-16	2.13636	1 L4	24	16	64	No	No	No	No	No	No	Yes	Yes
GCP	a2-ultragpu-1g	8.03933	1 A100	80	12	170	No	No	No	No	No	No	Yes	No
GCP	a3-highgpu-1g	16.71826	8 H100	640	208	1872	No	No	No	No	No	No	No	No
Azure	Standard_D4ds_v6	0.43947	-	-	4	16	No	No	No	No	No	Yes	No	No
Azure	Standard_D8ds_v6	0.89558	-	-	8	32	No	No	No	No	No	Yes	No	No
Azure	Standard_D16ds_v6	1.82258	-	-	16	64	No	No	No	No	No	Yes	No	No
Azure	Standard_D32ds_v6	3.47469	-	-	32	128	No	No	No	No	No	Yes	No	No

Provider	Instance Plan ID	Compute Factor	GPUs	GPU Memory (GB)	CPU Cores	Memory (GB)	AE01 UAE (Dubai)	EU01 Europe (Rotterdam)	EU02 Europe (Frankfurt)	EU10 Europe (Frankfurt)	EU11 Europe (Frankfurt)	EU20 Europe (Netherlands)	EU30 Europe (Frankfurt)	SA30 Saudi Arabia (Dammam)
Azure	Standard_NC4as_T4_v3	0.95221	1 T4	16	4	28	No	No	No	No	No	Yes	No	No
Azure	Standard_NC8as_T4_v3	1.43049	1 T4	16	8	56	No	No	No	No	No	Yes	No	No
Azure	Standard_NC16as_T4_v3	2.31391	1 T4	16	16	112	No	No	No	No	No	Yes	No	No
Azure	Standard_NC40ads_H100_v5	12.57589	1 H100	94	40	320	No	No	No	No	No	No	No	No
SAP	g_c4_m16	0.23058	-	-	4	16	Yes	Yes	Yes	No	No	No	No	No
SAP	g_c8_m32	0.46116	-	-	8	32	Yes	Yes	Yes	No	No	No	No	No
SAP	g_c32_m128	1.84472	-	-	32	128	No	Yes	Yes	No	No	No	No	No

Americas (AMER)

Provider	Instance Plan ID	Compute Factor	GPUs	GPU Memory (GB)	CPU Cores	Memory (GB)	BR10 Brazil (São Paulo)	CA10 Canada (Montreal)	US10 United States (Virginia)	US20 United States (West US)	US21 United States (East US)	US30 United States (Google Cloud)
AWS	m7i.xlarge	0.44536	-	-	4	16	Yes	Yes	Yes	No	No	No

Provider	Instance Plan ID	Compute Factor	GPUs	GPU Memory (GB)	CPU Cores	Memory (GB)	BR10 Brazil (São Paulo)	CA10 Canada (Montreal)	US10 United States (Virginia)	US20 United States (West US)	US21 United States (East US)	US30 United States (Google Cloud)
AWS	m7i.2xlarge	0.81903	-	-	8	32	Yes	Yes	Yes	No	No	No
AWS	m7i.4xlarge	1.56641	-	-	16	64	Yes	Yes	Yes	No	No	No
AWS	m7i.8xlarge	3.02526	-	-	32	128	Yes	Yes	Yes	No	No	No
AWS	g4dn.xlarge	0.98727	1 T4	16	4	16	Yes	Yes	Yes	No	No	No
AWS	g4dn.2xlarge	1.49228	1 T4	16	8	32	Yes	Yes	Yes	No	No	No
AWS	g4dn.4xlarge	2.36363	1 T4	16	16	64	Yes	Yes	Yes	No	No	No
AWS	g6.4xlarge	2.47566	1 L4	24	16	64	Yes	Yes	Yes	No	No	No
AWS	g6.12xlarge	8.07090	4 L4	96	96	384	No	Yes	Yes	No	No	No
AWS	g6e.4xlarge	5.33927	1 L40S	48	16	128	No	No	Yes	No	No	No
AWS	p4d.24xlarge	37.97723	8 A100	320	96	1152	No	No	No	No	No	No
GCP	n4-standard-4	0.37530	-	-	4	16	No	No	No	No	No	Yes
GCP	n4-standard-8	0.78152	-	-	8	32	No	No	No	No	No	Yes
GCP	n4-standard-16	1.50125	-	-	16	64	No	No	No	No	No	Yes
GCP	n4-standard-32	2.90973	-	-	32	128	No	No	No	No	No	Yes
GCP	n1-standard-4	0.36205	1 T4	16	4	15	No	No	No	No	No	Yes
GCP	n1-standard-8	0.72411	1 T4	16	8	30	No	No	No	No	No	Yes

Provider	Instance Plan ID	Compute Factor	GPUs	GPU Memory (GB)	CPU Cores	Memory (GB)	BR10 Brazil (São Paulo)	CA10 Canada (Montreal)	US10 United States (Virginia)	US20 United States (West US)	US21 United States (East US)	US30 United States (Google Cloud)
GCP	n1-standard-16	1.29359	1 T4	16	16	60	No	No	No	No	No	Yes
GCP	g2-standard-16	2.13636	1 L4	24	16	64	No	No	No	No	No	Yes
GCP	a2-ultragpu-1g	8.03933	1 A100	80	12	170	No	No	No	No	No	Yes
GCP	a3-high-gpu-1g	16.71826	8 H100	640	208	1872	No	No	No	No	No	Yes
Azure	Standard_D4ds_v6	0.43947	-	-	4	16	No	No	No	Yes	Yes	No
Azure	Standard_D8ds_v6	0.89558	-	-	8	32	No	No	No	Yes	Yes	No
Azure	Standard_D16ds_v6	1.82258	-	-	16	64	No	No	No	Yes	Yes	No
Azure	Standard_D32ds_v6	3.47469	-	-	32	128	No	No	No	Yes	Yes	No
Azure	Standard_NC4as_T4_v3	0.95221	1 T4	16	4	28	No	No	No	Yes	Yes	No
Azure	Standard_NC8as_T4_v3	1.43049	1 T4	16	8	56	No	No	No	Yes	Yes	No
Azure	Standard_NC16as_T4_v3	2.31391	1 T4	16	16	112	No	No	No	Yes	Yes	No
Azure	Standard_NC40ads_H100_v5	12.57589	1 H100	94	40	320	No	No	No	Yes	Yes	No

Provider	Instance Plan ID	Compute Factor	GPUs	GPU Memory (GB)	CPU Cores	Memory (GB)	BR10 Brazil (São Paulo)	CA10 Canada (Montreal)	US10 United States (Virginia)	US20 United States (West US)	US21 United States (East US)	US30 United States (Google Cloud)
SAP	g_c4_m16	0.23058	-	-	4	16	No	No	No	No	No	No
SAP	g_c8_m32	0.46116	-	-	8	32	No	No	No	No	No	No
SAP	g_c32_m128	1.84472	-	-	32	128	No	No	No	No	No	No

Conversion Rates for Node Hour Units

Node hour units represent the compute cost associated with a specific instance type. The number of node hour units consumed per hour depends on the instance type that you use. Refer to the conversion rates listed above to determine the node hour units consumed for each hour of instance usage.

Availability

SAP provisions requested instance types as quickly as possible. However, availability cannot be guaranteed. Provisioning delays may occur, or an instance type may be unavailable because of regional capacity or hyperscaler quota limitations. To avoid delays, plan for some flexibility and consider alternative instance types when appropriate.

1.3 Models

SAP AI Core provides access to a comprehensive collection of pre-trained machine learning models designed to accelerate your AI initiatives. These models serve as ready-to-use building blocks that can be integrated into your applications and business processes, eliminating the need to develop complex AI capabilities from scratch.

> **Note**
>
> SAP AI Core, including generative AI hub must **not** be used to generate synthetic data for training or fine-tuning models. Any exceptions must be explicitly approved and documented.

SAP AI Core Hosted Models

These are models that are deployed and managed on SAP AI Core infrastructure.

Open Source models are hosted by SAP AI Core and are accessible through an OpenAI-compatible API schema.

Remote Models

Remote models are hosted on remotely, on external platforms or infrastructures.

Models from Azure OpenAI are accessed through a private instance of the chat-completions API. These access points aren't publicly available and are accessed through SAP AI Core. For more information, see Chat completions in the Microsoft documentation.

Access to generative AI models falls under the global AI scenarios `foundation-models` and `orchestration`. SAP AI Core manages these scenarios. You can access individual models as executables through serving templates. To use a specific model, choose the corresponding template.

When to Use Orchestration vs. Foundation Models

Advantages of Using Orchestration

- Provider Agnostic: The orchestration service offers a unified interface for accessing LLMs from multiple providers, simplifying the process of testing and comparing models. You can switch models via a config parameter without creating new deployments.
- Simplified Integration: It streamlines the integration of LLMs into applications, ensuring efficient and cost-effective development.
- Expandability: The service allows you to start with basic features and add modules such as content filtering, anonymization, or grounding as needed, offering a smooth learning curve.

Considerations Before Using Orchestration

- Open Source Framework Support: While we support LangChain, a popular open-source framework, we can't support every framework available. If a specific open-source framework is required, you might need to extend it for support or consider alternative integration methods.

Workflow

You can access LLMs through the `foundation-models` and `orchestration` scenarios.

To use a model through the `foundation-models` scenario, you'll need to know its `executableId`. For more information about available models, region and cost information, and executable IDs, see Supported Models [page 50].

Next, create a configuration and deployment using the `foundation-models` and `executableId`. You'll then consume the model through API calls. For more information, see Create a Deployment and Foundation Models [page 190].

To use a model through the `orchestration` scenario, using the harmonized API, see Orchestration [page 210].

Related Information

Orchestration [page 210]
Foundation Models [page 190]

1.3.1 Supported Models

You can find information about available models, including token conversion rates, rate limits, and deprecation dates, in SAP Note 3437766.



Go to SAP Note 3437766.

- SAP Note 3437766 [https://me.sap.com/notes/3437766]

You can retrieve a list of available models and model versions using the model discovery endpoint. Send a GET request to endpoint `$AI_API_URL/v2/lm/scenarios/foundation-models/models`. The base URL is the URL of your SAP AI Core environment. This can also be set as an environment variable.

Include the following headers in your request:

- Authorization: Bearer <your-access-token>
- content-type: application/json
- AI-Resource-Group: <your-resource-group>

For example:

```
curl --location '$AI_API_URL/v2/lm/scenarios/foundation-models/models' \
--header 'AI-Resource-Group: <your-resource-group>' \
--header 'Authorization: Bearer $AUTH_TOKEN'
```

Example response

Output Code

```
{
  "count": integer,
  "resources": [
    {
      "accessType": "Remote",
      "allowedScenarios": [
        {
          "executableId": "<executable Id>",
          "scenarioId": "foundation-models"
        },
        {
          "executableId": "orchestration",
          "scenarioId": "orchestration"
        }
      ],
      "description": "<model description>",
      "displayName": "<model display name>",
      "executableId": "<executable ID>",
      "model": "<model ID>",
      "provider": "<Provider>",
      "versions": [
        {
          "capabilities": [
            "text-generation"
          ],
          "contextLength": integer,
          "cost": [
            {
              "inputCost": integer
            },
            {
              "outputCost": integer
            }
          ],
          "deprecated": Boolean,
          "inputTypes": [
            "text"
          ],
          "isLatest": Boolean,
          "metadata": [
            {
              "meanWinRate": integer
            },
            {
              "chatBotArenaScore": integer
            },
            {
              "helmCapabilitiesAccuracyMeanScore": integer
            }
          ],
          "name": "<name>",
          "retirementDate": "",
          "streamingSupported": Boolean
        }
      ]
    },
    # ...
  ]
}
```

The output can include the following parameters:

Parameter	Description
count	Number of models available
accessType	Type of access for the model (for example, "Remote")
allowedScenarios	List of scenarios and executables where the model can be used
description	Description of the model
displayName	Display name of the model
executableId	Executable under which the model is available, required at the time of creating Deployment configuration
model	Required as modelName at the time of creating Deployment configuration
provider	Provider of the model
versions	List of available model versions for given model
versions[].capabilities	List of capabilities of the model version
versions[].contextLength	Maximum context length supported by the model version
versions[].cost	List of cost information (for example, inputCost, outputCost)
versions[].deprecated	Boolean indicating if the model version is deprecated
versions[].inputTypes	List of supported input types
versions[].isLatest	Suggests if the modelVersion is latest
versions[].metadata	Additional metadata about the model version
versions[].name	Version name, optional value to provide as modelVersion at the time of creating Deployment Configuration
versions[].retirementDate	Date when the model version will be retired (if applicable)
versions[].streamingSupported	Boolean indicating if streaming is supported

For information about specific models and their parameters, see the documentation from the model providers.

Open Source models are hosted by SAP AI Core and are accessible through an OpenAI-compatible API schema.

1.3.2 AWS Bedrock

Context

You can access LLMs through the `foundation-models` and `orchestration` scenarios.

To use a model through the `orchestration` scenario, using the harmonized API, see Orchestration [page 210].

For more information about available models, including conversion rates for tokens, rate limits, and deprecation dates, see SAP Note 3437766.

Accessing Models Through Orchestration

Orchestration offers a harmonized API that allows you to use different models without changing the client code. You likely have an orchestration deployment running in your default resource group. If you want to access models through orchestration in a different resource group, create an orchestration deployment for your chosen resource group. For more information, see .Create a Deployment for Orchestration [page 703].

Access to orchestration of generative AI models is provided under the global AI scenario `orchestration`, which is managed by SAP AI Core.

To access generative AI models using orchestration, add the following information to the `model` module of your orchestration workflow:

- The name of your chosen model
- The version name of your chosen version. If no model version is listed, it is not applicable.

For more information about supported models and associated costs, see SAP Note 3505347

For more information about orchestration workflows, see Orchestration Workflow V2 [page 382].

Accessing Models through the `foundation-models` Scenario

You can access foundation models by creating a deployment for the model that you want to use. To do so, you need an auth token from your SAP AI Core instance. For more information, see Get an Auth Token [page 6] and Create a Deployment [page 194].

To create your deployment, you'll need the following information:

- The scenario: `foundation-models`
- The `executableId`: `aws-bedrock`
- The name of your chosen model
- The version name of your chosen version. If no model version is listed, it is not applicable.

To use a specific version of a model or to upgrade model versions manually, specify the version of your model deployment. To upgrade automatically, specify the model version as `latest`. For more information, see Model Lifecycle [page 194]. If a model version isn't listed, the model isn't applicable.

AWS Bedrock models are remote models.

After you've created a deployment for your model, you can consume the model using prompts. To access the model, you need your deployment ID, which you can set as an environment variable.

Check that you've set the following headers:

Header	Value
Authorization	Bearer $AUTH_TOKEN
AI-Resource-Group	The resource group used in the activation steps
$DEPLOYMENT_URL	The deployment URL for your generative AI model. For more information, see Create a Deployment [page 194]. Alternatively, you can replace the `$DEPLOYMENT_URL` placeholder in the curl command with your deployment URL.

Consumption

The following examples show how you can consume various generative AI models using curl. For more information about prompts, see the tutorial Prompt LLMs in the Generative AI Hub in SAP AI Core & Launchpad.

> → Tip
>
> If you use a Windows device, use Windows PowerShell, and replace `curl` with `curl.exe`.

For more information about supported parameters, see Supported Parameters [page 199].

Anthropic Claude

For more information from the model provider, see AWS Bedrock Claude .

Claude 3 Haiku | Claude 4 Sonnet | Claude 4 Opus | Claude 4.5 Haiku | Claude 4.5 Sonnet | Claude 4.6 Opus | Claude 4 Sonnet | Claude 4.7 Opus | Claude 4.8 Opus

Invoke:

```
curl --location '$DEPLOYMENT_URL/invoke' \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "anthropic_version": "bedrock-2023-05-31",
    "max_tokens": 100,
    "messages": [
        {
            "role": "user",
            "content": "Hello, Claude"
        }
    ]
}'
```

Converse:

```
curl --location '$DEPLOYMENT_URL/converse' \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
```

```
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "inferenceConfig": {
        "maxTokens": 100,
        "stopSequences": [
            "blab"
        ]
    },
    "messages": [
        {
            "content": [
                {
                    "text": "Perplexity means?"
                }
            ],
            "role": "user"
        }
    ]
}'
```

Amazon Titan

For more information from the model provider, see AWS Bedrock Titan .

Titan Embed Text

```
curl --location '$DEPLOYMENT_URL/invoke' \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "inputText": "Who im I?"
  }'
```

Titan Embed Image

```
curl --location '$DEPLOYMENT_URL/invoke' \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "inputText": "hi",
    "inputImage": "#this-is-an-example-base64-string-value for-an-image/9j/
4QDeRXhpZgAASUkqAAgAAAAGABIBAwABAAAAAQAAABoBBQABAAAAVgAAABsBBQABAAAAXgAAACgBAwABA
AAAAgAAABMCAwABAAAAAQAAAGmHBAABAAAAZgAAAAAAAABIAAAAAQAAAEgAAAABAAAABwAAkAcABAAAAD
AyMTABkQcABAAAAAECAwCGkgcAFgAAAMAAAAAAoAcABAAAADAxMDABoAMAAQAAAP//
AAACoAQAAQAAAMgAAAADoAQAAQAAAMgAAAAAAAAAQVNDSUkAAABQaWNzdW0gSUQ6IDEyM//
bAEMACAYGBwYFCAcHBwkJCAoMFA0MCwsMGRITDxQdGh8eHRocHCAkLicgIiwjHBwoNyksMDE0NDQfJzk9
ODI8LjM0Mv/bAEMBCQkJDAsMGA0NGDIhHCEyMjIyMjIyMjIyMjIyM",
    "embeddingConfig": {
        "outputEmbeddingLength": 256
    }
}'
```

Output Code

```
{
  "embedding": [
    -0.0013275146,
    -0.024658203,
```

```
      ...
      0.068359375,
      0.014404297,
      0.025512695
    ],
    "inputTextTokenCount": 3
  }
```

Amazon Nova

For more information from the model provider, see AWS Bedrock Nova .

Nova Pro | Nova Lite | Nova Micro | Nova Premier

```
curl --location '$DEPLOYMENT_URL/converse' \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "inferenceConfig": {
        "maxTokens": 100,
        "stopSequences": [
            "blab"
        ],
        "temperature": 0.7
    },
    "messages": [
        {
            "content": [
                {
                    "text": "Perplexity means?"
                }
            ],
            "role": "user"
        }
    ]
}'
```

Streaming

Where supported, streaming can be enabled for generative AI models to return responses incrementally.

For Amazon Bedrock models, streaming can be invoked by replacing the deployment URL with `$DEPLOYMENT_URL/invoke-with-response-stream`.

To enable streaming when using the Converse API, call the `$DEPLOYMENT_URL/converse-stream` endpoint instead of `/converse`.

Removing a Model

If you want to remove a model, delete its deployment. For more information, see Delete a Single Deployment

Related Information

Access the Streaming using the SAP AI SDK for Java
Access the Streaming using the SAP AI SDK for JavaScript

1.3.3 Cohere

Context

You can access LLMs through the `foundation-models` and `orchestration` scenarios.

To use a model through the `orchestration` scenario, using the harmonized API, see Orchestration [page 210].

For more information about available models, including conversion rates for tokens, rate limits, and deprecation dates, see SAP Note 3437766.

Accessing Models Through Orchestration

Orchestration offers a harmonized API that allows you to use different models without changing the client code. You likely have an orchestration deployment running in your default resource group. If you want to access models through orchestration in a different resource group, create an orchestration deployment for your chosen resource group. For more information, see .Create a Deployment for Orchestration [page 703].

Access to orchestration of generative AI models is provided under the global AI scenario `orchestration`, which is managed by SAP AI Core.

To access generative AI models using orchestration, add the following information to the `model` module of your orchestration workflow:

- The name of your chosen model
- The version name of your chosen version. If no model version is listed, it is not applicable.

For more information about supported models and associated costs, see SAP Note 3505347

For more information about orchestration workflows, see Orchestration Workflow V2 [page 382].

Accessing Models through the `foundation-models` Scenario

You can access foundation models by creating a deployment for the model that you want to use. To do so, you need an auth token from your SAP AI Core instance. For more information, see Get an Auth Token [page 6] and Create a Deployment [page 194].

To create your deployment, you'll need the following information:

- The scenario: `foundation-models`
- The `executableId`: `aicore-cohere`
- The name of your chosen model
- The version name of your chosen version. If no model version is listed, it is not applicable.

To use a specific version of a model or to upgrade model versions manually, specify the version of your model deployment. To upgrade automatically, specify the model version as `latest`. For more information, see Model Lifecycle [page 194]. If a model version isn't listed, the model isn't applicable.

For more information from the model provider, see Cohere Rerank .

After you've created a deployment for your model, you can consume the model using prompts. To access the model, you need your deployment ID, which you can set as an environment variable.

Check that you've set the following headers:

Header	Value
Authorization	Bearer $AUTH_TOKEN
AI-Resource-Group	The resource group used in the activation steps
$DEPLOYMENT_URL	The deployment URL for your generative AI model. For more information, see Create a Deployment [page 194]. Alternatively, you can replace the `$DEPLOYMENT_URL` placeholder in the curl command with your deployment URL.

Consumption

The following examples show how you can consume various generative AI models using curl. For more information about prompts, see the tutorial Prompt LLMs in the Generative AI Hub in SAP AI Core & Launchpad.

> → Tip
>
> If you use a Windows device, use Windows PowerShell, and replace `curl` with `curl.exe`.

For more information about supported parameters, see Supported Parameters [page 199].

Rerank

Cohere model is SAP managed model.

For more information from the model provider, see Cohere Rerank .

Rerank 3.5 |

```
curl --location "$DEPLOYMENT_URL/rerank" \
```

```
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "model": "cohere-reranker-35",
    "top_n": 3,
    "query": "What is the capital of Amesic?",
    "documents": [
        "Carson City is the capital city of the American state of Strita.",
        "The Commonwealth of the Northern Mariana Islands is a group of islands
in the Pacific Ocean. Its capital is Saipan.",
        "Anesia, D.C. is the capital of the Amesic States. It is a federal
district.",
        "Capitalization or capitalisation in English grammar is the use of a
capital letter at the start of a word. English usage varies from capitalization
in other languages.",
        "Capital punishment has existed in the Amesic since before it was a
country. As of 2017, capital punishment is legal in 30 of the 50 states."
    ]
}'
```

Output Code

```
{
  "id": "<id>",
  "meta": {
    "api_version": {
      "version": "1"
    },
    "billed_units": {
      "search_units": 1
    }
  },
  "results": [
    {
      "index": 2,
      "relevance_score": 0.85157084
    },
    {
      "index": 4,
      "relevance_score": 0.23100689
    },
    {
      "index": 1,
      "relevance_score": 0.056014005
    }
  ]
}
```

Reasoning

Command A Reasoning

Cohere model is SAP AI Core hosted.

For more information from the model provider, see Cohere Command A Reasoning .

```
curl --location "$DEPLOYMENT_URL/chat" \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
```

```
--data '{
    "model": "cohere--command-a-reasoning",
    "stream": false,
    "frequency_penalty": 0.8,
    "thinking": {
        "type": "enabled"
    },
    "messages": [
        {
            "role": "user",
            "content": "Tell me about the reflection?"
        }
    ]
}'
```

Removing a Model

If you want to remove a model, delete its deployment. For more information, see Delete a Single Deployment

1.3.4 Mistral

Context

You can access LLMs through the `foundation-models` and `orchestration` scenarios.

To use a model through the `orchestration` scenario, using the harmonized API, see Orchestration [page 210].

For more information about available models, including conversion rates for tokens, rate limits, and deprecation dates, see SAP Note 3437766.

Accessing Models Through Orchestration

Orchestration offers a harmonized API that allows you to use different models without changing the client code. You likely have an orchestration deployment running in your default resource group. If you want to access models through orchestration in a different resource group, create an orchestration deployment for your chosen resource group. For more information, see .Create a Deployment for Orchestration [page 703].

Access to orchestration of generative AI models is provided under the global AI scenario `orchestration`, which is managed by SAP AI Core.

To access generative AI models using orchestration, add the following information to the `model` module of your orchestration workflow:

- The name of your chosen model
- The version name of your chosen version. If no model version is listed, it is not applicable.

For more information about supported models and associated costs, see SAP Note 3505347

For more information about orchestration workflows, see Orchestration Workflow V2 [page 382].

Accessing Models through the `foundation-models` Scenario

You can access foundation models by creating a deployment for the model that you want to use. To do so, you need an auth token from your SAP AI Core instance. For more information, see Get an Auth Token [page 6] and Create a Deployment [page 194].

To create your deployment, you'll need the following information:

- The scenario: `foundation-models`
- The `executableId`: `aicore-mistralai`
- The name of your chosen model
- The version name of your chosen version. If no model version is listed, it is not applicable.

To use a specific version of a model or to upgrade model versions manually, specify the version of your model deployment. To upgrade automatically, specify the model version as `latest`. For more information, see Model Lifecycle [page 194]. If a model version isn't listed, the model isn't applicable.

Mistral models are SAP AI Core hosted.

For more information from the model provider, see Mistral Large and Mistral Small .

After you've created a deployment for your model, you can consume the model using prompts. To access the model, you need your deployment ID, which you can set as an environment variable.

Check that you've set the following headers:

Header	Value
Authorization	Bearer $AUTH_TOKEN
AI-Resource-Group	The resource group used in the activation steps
$DEPLOYMENT_URL	The deployment URL for your generative AI model. For more information, see Create a Deployment [page 194]. Alternatively, you can replace the `$DEPLOYMENT_URL` placeholder in the curl command with your deployment URL.

Consumption

The following examples show how you can consume various generative AI models using curl. For more information about prompts, see the tutorial Prompt LLMs in the Generative AI Hub in SAP AI Core & Launchpad.

→ Tip

If you use a Windows device, use Windows PowerShell, and replace `curl` with `curl.exe`.

For more information about supported parameters, see Supported Parameters [page 199].

mistralai--mistral-large-instruct | mistralai--mistral-small-instruct| mistralai--mistral-small

Text input

Without streaming:

```
curl --location '$DEPLOYMENT_URL/chat/completions' \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "model": "<modelName>",
    "messages": [
        {
            "role": "user",
            "content": "Whats the difference between Accountability vs
Responsibility, answer in 200 words?"
        }
    ],
    "max_tokens": 100
}'
```

{·} Output Code


```
{
  "choices": [
    {
      "finish_reason": "length",
      "index": 0,
      "logprobs": null,
      "message": {
        "annotations": null,
        "audio": null,
        "content": "**Accountability vs. Responsibility**\n\nWhile often used
interchangeably, **responsibility** and **accountability** have distinct
meanings in personal and professional contexts.\n\n**Responsibility** refers
to the obligation to perform a task or role. It is about **what** you are
expected to do—your duties, tasks, or assignments. For example, a manager
is *responsible* for leading a team or overseeing a project. Responsibility
is assigned (e.g., job descriptions, contracts) and does not necessarily
imply consequences for outcomes.\n\n**Accountability**, on the other hand, is
about **answering for results**. It involves taking ownership of outcomes,
whether successful or not, and being prepared to explain, justify, or
face consequences. Accountability requires transparency, integrity, and a
willingness to learn from mistakes. For instance, if a project fails, an
*accountable* person does not shift blame but explains what went wrong and
how to improve.\n\nIn essence:\n- **Responsibility** = *Doing* the work.\n-
**Accountability** = *Answering* for the work's results.\n\nA person can
be *responsible* without being *accountable* if they delegate tasks without
oversee",
        "function_call": null,
        "reasoning": null,
```

```
        "refusal": null,
        "role": "assistant",
        "tool_calls": []
      },
      "stop_reason": null,
      "token_ids": null
    }
  ],
  "created": 1777541318,
  "id": "chatcmpl-963fc89d2548ada6",
  "kv_transfer_params": null,
  "model": "mistralai--mistral-small",
  "object": "chat.completion",
  "prompt_logprobs": null,
  "prompt_token_ids": null,
  "service_tier": null,
  "system_fingerprint": null,
  "usage": {
    "completion_tokens": 250,
    "prompt_tokens": 30,
    "prompt_tokens_details": null,
    "total_tokens": 280
  }
}
```


With streaming:

```
curl --location '$DEPLOYMENT_URL/chat/completions' \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "model": "<ModelName>",
    "messages": [
        {
            "role": "user",
            "content": "Sample prompt"
        }
    ],
    "max_tokens": 100,
    "stream": true
}'
```

Image input (for mistralai--mistral-small-instruct only):

```
curl --location '$DEPLOYMENT_URL/chat/completions' \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "model": "<modelName>",
    "messages": [
        {
            "role": "user",
            "content": [
                {
                    "type": "image_url",
                    "image_url": {
                        "url": "https://url/pexels-field-1-1024×683.jpg"
                    }
                }
            ]
        }
    ],
    "max_tokens": 10
```

```
}'
```

Output Code


```
{
  "choices": [
    {
      "finish_reason": "length",
      "index": 0,
      "logprobs": null,
      "message": {
        "content": "The image shows a picturesque countryside landscape,
likely",
        "reasoning_content": null,
        "role": "assistant",
        "tool_calls": []
      },
      "stop_reason": null
    }
  ],
  "created": 1748573681,
  "id": "<id>",
  "model": "<modelName>",
  "object": "chat.completion",
  "prompt_logprobs": null,
  "usage": {
    "completion_tokens": 10,
    "prompt_tokens": 953,
    "prompt_tokens_details": null,
    "total_tokens": 963
  }
}
```


Note

For information on metering of image input, see Metering and Pricing for Generative AI.

mistralai--mistral-medium-instruct

For more information from the model provider, see Mistral Models .

Text Generation

```
curl --location '$DEPLOYMENT_URL/chat/completions' \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "model": "mistralai--mistral-medium-instruct",
    "messages": [
            {
              "role": "user",
              "content": "Give me a JSON object with a person's name and age."
            }
    ],
    "response_format": {
      "type": "json_object"
    }
```

```
}'
```

Output Code


```
{
  "id": "<id>",
  "object": "chat.completion",
  "created": 1757916217,
  "model": "mistralai--mistral-medium-instruct",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "reasoning_content": null,
        "content": "{\"name\": \"John Doe\", \"age\": 30}",
        "tool_calls": []
      },
      "logprobs": null,
      "finish_reason": "stop",
      "stop_reason": null
    }
  ],
  "usage": {
    "prompt_tokens": 16,
    "total_tokens": 31,
    "completion_tokens": 15,
    "prompt_tokens_details": null
  },
  "prompt_logprobs": null
}
```


Tool Calling

```
curl --location '$DEPLOYMENT_URL/chat/completions' \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "model": "mistralai--mistral-medium-instruct",
    "messages": [
        {
            "role": "system",
            "content": "You are a helpful assistant."
        },
        {
            "role": "user",
            "content": "What is the temperature in Boston today? Tell me in
fahrenreit."
        }
    ],
    "tools": [
        {
            "type": "function",
            "function": {
                "name": "get_current_weather",
                "description": "Get the current weather in a given location",
                "parameters": {
                    "type": "object",
                    "properties": {
                        "location": {
                            "type": "string",
                            "description": "The city and state, e.g. San
Francisco, CA"
                        },
```

```
                        "unit": {
                            "type": "string",
                            "enum": [
                                "celsius",
                                "fahrenheit"
                            ]
                        }
                    },
                    "required": [
                        "location"
                    ]
                }
            }
        }
    ],
    "temperature": 0.1,
    "top_p": 0.8,
    "max_tokens": 512
}'
```

Output Code

```
{
  "id": "<id>",
  "object": "chat.completion",
  "created": 1757916263,
  "model": "mistralai--mistral-medium-instruct",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "reasoning_content": null,
        "content": null,
        "tool_calls": [
          {
            "id": "RluueJ6B5",
            "type": "function",
            "function": {
              "name": "get_current_weather",
              "arguments": "{\"location\": \"Boston, MA\", \"unit\":
 \"fahrenheit\"}"
            }
          }
        ]
      },
      "logprobs": null,
      "finish_reason": "stop",
      "stop_reason": null
    }
  ],
  "usage": {
    "prompt_tokens": 126,
    "total_tokens": 148,
    "completion_tokens": 22,
    "prompt_tokens_details": null
  },
  "prompt_logprobs": null
}
```

Removing a Model

If you want to remove a model, delete its deployment. For more information, see Delete a Single Deployment

1.3.5 NVIDIA

Context

You can access LLMs through the `foundation-models` and `orchestration` scenarios.

To use a model through the `orchestration` scenario, using the harmonized API, see Orchestration [page 210].

For more information about available models, including conversion rates for tokens, rate limits, and deprecation dates, see SAP Note 3437766.

Accessing Models Through Orchestration

Orchestration offers a harmonized API that allows you to use different models without changing the client code. You likely have an orchestration deployment running in your default resource group. If you want to access models through orchestration in a different resource group, create an orchestration deployment for your chosen resource group. For more information, see .Create a Deployment for Orchestration [page 703].

Access to orchestration of generative AI models is provided under the global AI scenario `orchestration`, which is managed by SAP AI Core.

To access generative AI models using orchestration, add the following information to the `model` module of your orchestration workflow:

- The name of your chosen model
- The version name of your chosen version. If no model version is listed, it is not applicable.

For more information about supported models and associated costs, see SAP Note 3505347

For more information about orchestration workflows, see Orchestration Workflow V2 [page 382].

Accessing Models through the `foundation-models` Scenario

You can access foundation models by creating a deployment for the model that you want to use. To do so, you need an auth token from your SAP AI Core instance. For more information, see Get an Auth Token [page 6] and Create a Deployment [page 194].

To create your deployment, you'll need the following information:

- The scenario: `foundation-models`
- The `executableId`: `aicore-nvidia`
- The name of your chosen model
- The version name of your chosen version. If no model version is listed, it is not applicable.

To use a specific version of a model or to upgrade model versions manually, specify the version of your model deployment. To upgrade automatically, specify the model version as `latest`. For more information, see Model Lifecycle [page 194]. If a model version isn't listed, the model isn't applicable.

NVIDIA models are SAP managed.

For more information from the model provider, see NVIDIA .

After you've created a deployment for your model, you can consume the model using prompts. To access the model, you need your deployment ID, which you can set as an environment variable.

Check that you've set the following headers:

Header	Value
Authorization	Bearer $AUTH_TOKEN
AI-Resource-Group	The resource group used in the activation steps
$DEPLOYMENT_URL	The deployment URL for your generative AI model. For more information, see Create a Deployment [page 194]. Alternatively, you can replace the `$DEPLOYMENT_URL` placeholder in the curl command with your deployment URL.

Consumption

The following examples show how you can consume various generative AI models using curl. For more information about prompts, see the tutorial Prompt LLMs in the Generative AI Hub in SAP AI Core & Launchpad.

> → Tip
>
> If you use a Windows device, use Windows PowerShell, and replace `curl` with `curl.exe`.

For more information about supported parameters, see Supported Parameters [page 199].

Llama

nvidia--llama-3.2-nv-embedqa-1b

```
curl --location '$DEPLOYMENT_URL/embeddings' \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
```

```
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "input": ["Hello world"],
    "model": "nvidia--llama-3.2-nv-embedqa-1b",
    "input_type": "query",
    "encoding_format": "float",
    "truncate": "NONE"
}'
```

Removing a Model

If you want to remove a model, delete its deployment. For more information, see Delete a Single Deployment

1.3.6 OpenAI

Context

You can access LLMs through the `foundation-models` and `orchestration` scenarios.

To use a model through the `orchestration` scenario, using the harmonized API, see Orchestration [page 210].

For more information about available models, including conversion rates for tokens, rate limits, and deprecation dates, see SAP Note 3437766.

Accessing Models Through Orchestration

Orchestration offers a harmonized API that allows you to use different models without changing the client code. You likely have an orchestration deployment running in your default resource group. If you want to access models through orchestration in a different resource group, create an orchestration deployment for your chosen resource group. For more information, see .Create a Deployment for Orchestration [page 703].

Access to orchestration of generative AI models is provided under the global AI scenario `orchestration`, which is managed by SAP AI Core.

To access generative AI models using orchestration, add the following information to the `model` module of your orchestration workflow:

- The name of your chosen model
- The version name of your chosen version. If no model version is listed, it is not applicable.

For more information about supported models and associated costs, see SAP Note 3505347

For more information about orchestration workflows, see Orchestration Workflow V2 [page 382].

Accessing Models through the `foundation-models` Scenario

You can access foundation models by creating a deployment for the model that you want to use. To do so, you need an auth token from your SAP AI Core instance. For more information, see Get an Auth Token [page 6] and Create a Deployment [page 194].

To create your deployment, you'll need the following information:

- The scenario: `foundation-models`
- The `executableId`: `azure-openai`
- The name of your chosen model
- The version name of your chosen version. If no model version is listed, it is not applicable.

To use a specific version of a model or to upgrade model versions manually, specify the version of your model deployment. To upgrade automatically, specify the model version as `latest`. For more information, see Model Lifecycle [page 194]. If a model version isn't listed, the model isn't applicable.

Open AI models are remote models.

After you've created a deployment for your model, you can consume the model using prompts. To access the model, you need your deployment ID, which you can set as an environment variable.

Check that you've set the following headers:

Header	Value
Authorization	Bearer $AUTH_TOKEN
AI-Resource-Group	The resource group used in the activation steps
$DEPLOYMENT_URL	The deployment URL for your generative AI model. For more information, see Create a Deployment [page 194]. Alternatively, you can replace the `$DEPLOYMENT_URL` placeholder in the curl command with your deployment URL.

Consumption

The following examples show how you can consume various generative AI models using curl. For more information about prompts, see the tutorial Prompt LLMs in the Generative AI Hub in SAP AI Core & Launchpad.

> → Tip
>
> If you use a Windows device, use Windows PowerShell, and replace `curl` with `curl.exe`.

For more information about supported parameters, see Supported Parameters [page 199].

Completions

For more information from the model provider, see Microsoft Azure Open AI Chat Completions .

o4-mini | o3 | gpt-4.1 | gpt-4.1-mini | gpt-4.1-nano | gpt-5 | gpt-5-mini | gpt-5-nano | gpt-5.1 | gpt-5.2 | gpt-5.4 | gpt-5.4-nano | gpt-5.5 | gpt-5.6-sol | gpt-5.6-terra | gpt-5.6-luna | gpt-6.1-sol | gpt-6-astra | gpt-6-luna

Text Input

```
curl --location '$DEPLOYMENT_URL/chat/completions?api-version=2023-05-15' \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "messages": [
        {
            "role": "user",
            "content": "sample input prompt"
        }
    ],
    "max_tokens": 100,
    "temperature": 0,
    "frequency_penalty": 0,
    "presence_penalty": 0,
    "stop": "null"
}'
```

o1 | o3-mini

```
curl --location '$DEPLOYMENT_URL/chat/completions?api-version=2024-12-01-
preview' \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "messages": [
      {
        "role": "user",
        "content": "Hello!"
      }
    ]
  }'
```

GPT-4o | GPT-4o Mini

Image input

```
curl --location '$DEPLOYMENT_URL/chat/completions?api-version=2023-05-15' \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "messages": [
        {
            "role": "user",
            "content": [
                {
                    "type": "text",
                    "text": "Describe this picture:"
                },
                {
```

```
                    "type": "image_url",
                    "image_url": {
                        "url": "https://path/images/image.png"
                    }
                }
            ]
        }
    ],
    "max_tokens": 10
}'
```

Embeddings

For more information from the model provider, see Microsoft Azure Open AI Embeddings .

text-embedding-ada-002 | text-embedding-3-small | text-embedding-3-large

```
curl --location '$DEPLOYMENT_URL/embeddings?api-version=2023-05-15' \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "input": "sample input prompt"
}'
```

Realtime

For more information from the model provider, see Microsoft Azure OpenAI Realtime Audio .

`gpt-realtime` can be accessed via WebSockets, via any programming language that supports WebSockets.

The following example uses Python language and the WebSockets library to interact with the `gpt-realtime` model.

Before running your python script, you must complete the following steps:

1. Create a Python virtual environment.

   ```
   python3 -m venv .venv
   source .venv/bin/activate
   ```

2. Install the OpenAI Python client library.

   ```
   pip3 install 'openai[realtime]' 'sounddevice'
   ```

3. Set the required environment variables.

   ```
   export BEARER_TOKEN='<yourToken>'
   export DEPLOYMENT_URL='<yourDeploymentUrl'
   export RESOURCE_GROUP='<yourResourceGroup>'
   ```

4. To use the model with **audio** input, run the script with the following parameters:

   ```
   python3 speech-to-speech.py
   ```

To use the model with **text** input, run the script with the following parameters:

```
python3 speech-to-speech.py --mode text
```

You can use the Realtime API via WebSockets to send text or audio input to the model and receive audio output in real time. The Realtime API via WebRTC is not supported.

Speech to Speech

The following example uses the OpenAI client library to interact with the gpt-realtime model via WebSockets.

```
 import argparse
import asyncio
import base64
import json
import os
import sounddevice as sd
import websockets
from websockets.exceptions import ConnectionClosedOK, ConnectionClosedError
parser = argparse.ArgumentParser(description='GPT Realtime API Client')
parser.add_argument('--mode', choices=['audio', 'text'], default='audio',
                    help='Input mode: audio (microphone) or text (keyboard).
Default: audio')
args = parser.parse_args()
INPUT_MODE = args.mode
SAMPLE_RATE = 24000
CHUNK_SIZE = 8192
AUDIO_FORMAT = 'int16'
BEARER_TOKEN = os.getenv("BEARER_TOKEN")
RESOURCE_GROUP = os.getenv("RESOURCE_GROUP")
DEPLOYMENT_URL = os.getenv("DEPLOYMENT_URL")
WEBSOCKET_URL = f"{DEPLOYMENT_URL}/v1/realtime"
# Print control configuration
PRINT_CONFIG = {
    "usage_metadata": True,
    "session_updates": True,
    "debug": True
}
async def _check_for_exit():
    """Monitor for exit command input"""
    while True:
        try:
            line = await asyncio.to_thread(input, "")
            if line.strip().lower() == "exit":
                return True
        except (EOFError, KeyboardInterrupt):
            return True
        await asyncio.sleep(0.1)
async def audio_player(queue: asyncio.Queue):
    with sd.RawOutputStream(samplerate=SAMPLE_RATE, channels=1,
dtype=AUDIO_FORMAT) as audio_out:
        while True:
            pcm = await queue.get()
            if pcm is None:
                break
            audio_out.write(pcm)
            queue.task_done()
async def receive_messages(ws, audio_queue, response_complete_event=None):
    """Receive and process WebSocket messages"""
    try:
        async for message in ws:
            data = json.loads(message)
            msg_type = data.get("type")
            if msg_type == "session.created":
                session_id = data.get("session", {}).get("id", "unknown")
                if PRINT_CONFIG["session_updates"]:
```

```
                    print(f"Connected. Session ID: {session_id}")
            elif msg_type == "session.updated":
                if PRINT_CONFIG["session_updates"]:
                    print(f"Session updated: {data.get('session', {})}")
            # User transcript (transcribes what you said but for audio mode only)
            elif msg_type ==
"conversation.item.input_audio_transcription.completed":
                transcript = data.get("transcript", "")
                if transcript and INPUT_MODE == "audio":
                    print(f"You said: {transcript}")
            # Model transcript (streamed in real time)
            elif msg_type == "response.output_audio_transcript.delta":
                delta = data.get("delta", "")
                if delta:
                    print(delta, end="", flush=True)
            elif msg_type == "response.output_audio_transcript.done":
                print()  # finish transcript line
            elif msg_type == "response.done":
                response_data = data.get("response", {})
                usage = response_data.get("usage")
                if usage and PRINT_CONFIG["usage_metadata"]:
                    print(f"Usage metadata: {usage}")
                # Signal that response is complete (for text mode)
                if response_complete_event:
                    response_complete_event.set()
            # Model's audio playback
            elif msg_type == "response.output_audio.delta":
                audio_b64 = data.get("delta")
                if audio_b64:
                    pcm = base64.b64decode(audio_b64)
                    await audio_queue.put(pcm)
            elif msg_type == "error":
                error_data = data.get("error", {})
                error_type = error_data.get('type', 'unknown')
                error_msg = error_data.get('message', 'No message')
                print(f"\n[ERROR] {error_type}: {error_msg}")
                if PRINT_CONFIG["debug"]:
                    print(f"\n[DEBUG] Full error data: {json.dumps(data,
indent=2)}")
    except ConnectionClosedOK:
        pass
    except ConnectionClosedError as e:
        if e.code is None or e.code == 1006:
            print("\n" + "="*80)
            print("ERROR: Connection closed unexpectedly")
            print("="*80)
        else:
            print(f"\nConnection closed: Code {e.code} - {e.reason}")
    except Exception as e:
        print(f"Receive error: {e}")
async def text_input_sender(ws, response_complete_event):
    """Send text messages from keyboard input"""
    print("\nType your message and press Enter. Type 'exit' to quit.\n")
    while True:
        try:
            user_text = await asyncio.to_thread(input, "You: ")
            if user_text.strip().lower() == "exit":
                print("Exit requested, ending session...")
                break
            if not user_text.strip():
                continue
            # Clear the response complete flag
            response_complete_event.clear()
            print("Assistant: ", end="", flush=True)
            text_message = {
                "type": "conversation.item.create",
                "item": {
                    "type": "message",
```

```
                    "role": "user",
                    "content": [
                        {
                            "type": "input_text",
                            "text": user_text
                        }
                    ]
                }
            }
            await ws.send(json.dumps(text_message))
            # Request response
            response_request = {
                "type": "response.create"
            }
            await ws.send(json.dumps(response_request))
            # Wait for response to complete before prompting again
            await response_complete_event.wait()
        except (EOFError, KeyboardInterrupt):
            print("\nKeyboard interrupt received, ending session...")
            break
        except Exception as e:
            print(f"\nInput error: {e}")
            break
async def main():
    print("Starting GPT Realtime API session...")
    audio_queue = asyncio.Queue()
    player_task = asyncio.create_task(audio_player(audio_queue))
    headers = {
        "AI-Resource-Group": RESOURCE_GROUP,
        "Authorization": f"Bearer {BEARER_TOKEN}",
    }
    session_config = {
        "type": "realtime",
        "output_modalities": ["audio"],
        "instructions": "Only respond in English. You are a helpful assistant.",
        "audio": {
            "input": {
                "format": {
                    "type": "audio/pcm",
                    "rate": 24000
                }
            },
            "output": {
                "format": {
                    "type": "audio/pcm",
                    "rate": 24000
                },
                "voice": "alloy"
            }
        }
    }
    # Add transcription for audio mode
    if INPUT_MODE == "audio":
        session_config["audio"]["input"]["transcription"] = {
            "model": "whisper-1"
        }
        session_config["audio"]["input"]["turn_detection"] = {
            "type": "server_vad",
            "threshold": 0.5,
            "prefix_padding_ms": 300,
            "silence_duration_ms": 200,
            "create_response": True
        }
    setup_msg = {
        "type": "session.update",
        "session": session_config
    }
    try:
```

```
        async with websockets.connect(WEBSOCKET_URL, additional_headers=headers)
as ws:
            print("WebSocket connection successful")
            await ws.send(json.dumps(setup_msg))
            receiver_task = asyncio.create_task(receive_messages(ws,
audio_queue))
            # Wait a bit for session to be established
            await asyncio.sleep(1)
            if INPUT_MODE == "audio":
                exit_task = asyncio.create_task(_check_for_exit())
                with sd.RawInputStream(
                    samplerate=SAMPLE_RATE,
                    blocksize=CHUNK_SIZE,
                    channels=1,
                    dtype=AUDIO_FORMAT
                ) as audio_in:
                    try:
                        print("Real time audio is currently being captured. Type
'exit' to quit.")
                        while not exit_task.done():
                            # Check if receiver task is still running
                            if receiver_task.done():
                                break
                            audio_chunk, overflowed = audio_in.read(CHUNK_SIZE)
                            chunk_b64 =
base64.b64encode(audio_chunk).decode('ascii')
                            frame = {"type": "input_audio_buffer.append",
"audio": chunk_b64}
                            try:
                                await ws.send(json.dumps(frame))
                            except Exception as e:
                                print(f"Send error: {e!r}")
                                break
                            await asyncio.sleep(0.01)
                        if exit_task.done():
                            print("Exit requested, ending session...")
                    except KeyboardInterrupt:
                        print("\nKeyboard interrupt received, ending session...")
                    finally:
                        # Clean up tasks
                        if not exit_task.done():
                            exit_task.cancel()
                        if not receiver_task.done():
                            receiver_task.cancel()
                        # Stop audio player
                        await audio_queue.put(None)
                        await player_task
                        print("Session closed.")
            else:
                print(f"Text mode active. Audio responses will play through
speakers.")
                try:
                    # Create event to signal when response is complete
                    response_complete_event = asyncio.Event()
                    response_complete_event.set()  # Initially set so first
input doesn't wait
                    # Update receiver task to pass the event
                    receiver_task.cancel()
                    receiver_task = asyncio.create_task(receive_messages(ws,
audio_queue, response_complete_event))
                    text_sender_task = asyncio.create_task(text_input_sender(ws,
response_complete_event))
                    # Wait for text sender or receiver to complete
                    done, pending = await asyncio.wait(
                        [text_sender_task, receiver_task],
                        return_when=asyncio.FIRST_COMPLETED
                    )
                    for task in pending:
```

```
                        task.cancel()
                        try:
                            await task
                        except asyncio.CancelledError:
                            pass
                except KeyboardInterrupt:
                    print("\nKeyboard interrupt received, ending session...")
                finally:
                    # Stop audio player
                    await audio_queue.put(None)
                    await player_task
                    print("Session closed.")
    except websockets.exceptions.InvalidStatus as e:
        # Handle HTTP status code errors during connection
        status = e.response.status_code
        print("\n" + "="*80)
        print(f"ERROR: Connection Failed (HTTP {status})")
        print("="*80)
        print(f"\n{e}")
        print(f"\nCurrent configuration:")
        print(f"  URL: {WEBSOCKET_URL}")
        print(f"  Resource Group: {RESOURCE_GROUP}")
    except websockets.exceptions.WebSocketException as e:
        print(f"\n" + "="*80)
        print(f"ERROR: WebSocket Error")
        print("="*80)
        print(f"\n{e}")
        print("\nPossible causes:")
        print("  - Network connectivity issues")
        print("  - Endpoint is unreachable")
        print("  - Firewall or proxy blocking connection")
    except Exception as e:
        print(f"\n" + "="*80)
        print("ERROR: Unexpected Error")
        print("="*80)
        print(f"\n{type(e).__name__}: {e}")
        if PRINT_CONFIG["debug"]:
            import traceback
            traceback.print_exc()
    finally:
        # Ensure audio player is stopped
        await audio_queue.put(None)
        await player_task
if __name__ == "__main__":
    asyncio.run(main())
```

When you want to end your session, you can close the connection by:

- Running `exit`
- Pressing `CTRL` + `C`

Removing a Model

If you want to remove a model, delete its deployment. For more information, see Delete a Single Deployment

Related Information

Access the Chat Completion using the SAP AI SDK for Java

Access the Chat Completion using the SAP AI SDK for JavaScript

Access the Embedding using the SAP AI SDK for Java

Access the Embedding using the SAP AI SDK for JavaScript

1.3.6.1 Responses API

The Responses API combines the simplicity of the Chat Completions API with the advanced tool-calling capabilities of the legacy Assistants API. It provides a streamlined way to build powerful agentic experiences by enabling developers to structure prompts, invoke tools, and manage outputs within a single API call.

Prerequisites

- Ensure that you have a deployment in the RUNNING state for a supported model and access to the DEPLOYMENT_URL. For more information, see Create a Deployment [page 194].
  Not all models support response API. Supported models are listed in the **SAP Note** 3437766 under the section **Models supported for Responses API**.

Response API Reference

For the complete OpenAI Responses API reference, see the Azure Docs on GitHub: https://github.com/Azure/azure-rest-api-specs/blob/main/specification/ai/data-plane/OpenAI.v1/azure-v1-v1-generated.yaml .

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $AUTH_TOKEN
AI-Resource-Group	The resource group used in the activation steps
$DEPLOYMENT_URL	The deployment URL for your generative AI model. For more information, see Create a Deployment [page 194]. Alternatively, you can replace the `$DEPLOYMENT_URL` placeholder in the curl command with your deployment URL.

1.3.6.1.1 Supported Consumption Methods

The following features are supported through SAP AI Core. Other features cited in the Azure Open AI documentation are not supported.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $AUTH_TOKEN
AI-Resource-Group	The resource group used in the activation steps
$DEPLOYMENT_URL	The deployment URL for your generative AI model. For more information, see Create a Deployment [page 194]. Alternatively, you can replace the `$DEPLOYMENT_URL` placeholder in the curl command with your deployment URL.

> **Note**
>
> By default, the **`store`** parameter is set to `<false>`, and responses are not stored.
>
> Optionally, you can set **`store`** to `<true>` and responses could then be stored by Microsoft for up to 30 days. Saved requests can be deleted by sending a DELETE request to the endpoint `/v1/responses/<response_id>`.

Basic Text Generation

Sample Code

```
curl --location conkeyref="$DEPLOYMENT_URL/responses" \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "model": "gpt-5",
  "input": "Count to 3 in Spanish."
}'
```

Output Code

```
{
  "background": false,
  "completed_at": 1775625426,
  "content_filters": [
    {
      "blocked": false,
      "content_filter_offsets": {
        "check_offset": 0,
        "end_offset": 1152,
        "start_offset": 1130
      },
      "content_filter_raw": [],
      "content_filter_results": {
        "hate": {
          "filtered": false,
          "severity": "safe"
        },
        "self_harm": {
          "filtered": false,
          "severity": "safe"
        },
        "sexual": {
```

```
          "filtered": false,
          "severity": "safe"
        },
        "violence": {
          "filtered": false,
          "severity": "safe"
        }
      },
      "source_type": "prompt"
    },
    {
      "blocked": false,
      "content_filter_offsets": {
        "check_offset": 0,
        "end_offset": 480,
        "start_offset": 0
      },
      "content_filter_raw": [],
      "content_filter_results": {
        "hate": {
          "filtered": false,
          "severity": "safe"
        },
        "self_harm": {
          "filtered": false,
          "severity": "safe"
        },
        "sexual": {
          "filtered": false,
          "severity": "safe"
        },
        "violence": {
          "filtered": false,
          "severity": "safe"
        }
      },
      "source_type": "completion"
    }
  ],
  "created_at": 1775625423,
  "error": null,
  "frequency_penalty": 0.0,
  "id": "resp_004fb524e4fff7f40069d5e4cfa8448195bf9627c7360e98a4",
  "incomplete_details": null,
  "instructions": null,
  "max_output_tokens": null,
  "max_tool_calls": null,
  "metadata": {},
  "model": "gpt-5",
  "object": "response",
  "output": [
    {
      "id": "rs_004fb524e4fff7f40069d5e4d045ec8195a5977f4fb6ed97c0",
      "summary": [],
      "type": "reasoning"
    },
    {
      "content": [
        {
          "annotations": [],
          "logprobs": [],
          "text": "Uno, dos, tres.",
          "type": "output_text"
        }
      ],
      "id": "msg_004fb524e4fff7f40069d5e4d1b6bc8195b5d071ab98fbfdda",
      "role": "assistant",
      "status": "completed",
```

```
      "type": "message"
    }
  ],
  "parallel_tool_calls": true,
  "presence_penalty": 0.0,
  "previous_response_id": null,
  "prompt_cache_key": null,
  "prompt_cache_retention": null,
  "reasoning": {
    "effort": "medium",
    "summary": null
  },
  "safety_identifier": null,
  "service_tier": "auto",
  "status": "completed",
  "store": true,
  "temperature": 1.0,
  "text": {
    "format": {
      "type": "text"
    },
    "verbosity": "medium"
  },
  "tool_choice": "auto",
  "tools": [],
  "top_logprobs": 0,
  "top_p": 1.0,
  "truncation": "disabled",
  "usage": {
    "input_tokens": 13,
    "input_tokens_details": {
      "cached_tokens": 0
    },
    "output_tokens": 143,
    "output_tokens_details": {
      "reasoning_tokens": 128
    },
    "total_tokens": 156
  },
  "user": null
}
```

Stream Text Generation

Sample Code

```
curl --location "$DEPLOYMENT_URL/responses" \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "model": "gpt-5",
  "input": "Tell me a short joke.",
  "stream": true
}'
```

Output Code

```
data: {"type":"response.created","response":
{"id":"resp_0f2d32831fb00e2e0069d5f8e804808193b3379a80c7a3b172","object":"resp
```

```
onse","created_at":1775630568,"status":"in_progress","background":false,"compl
eted_at":null,"content_filters":null,"error":null,"frequency_penalty":0.0,"inc
omplete_details":null,"instructions":null,"max_output_tokens":null,"max_tool_c
alls":null,"model":"gpt-5","output":
[],"parallel_tool_calls":true,"presence_penalty":0.0,"previous_response_id":nu
ll,"prompt_cache_key":null,"prompt_cache_retention":null,"reasoning":
{"effort":"medium","summary":null},"safety_identifier":null,"service_tier":"au
to","store":true,"temperature":1.0,"text":{"format":
{"type":"text"},"verbosity":"medium"},"tool_choice":"auto","tools":
[],"top_logprobs":0,"top_p":1.0,"truncation":"disabled","usage":null,"user":nu
ll,"metadata":{}},"sequence_number":0}
data: {"type":"response.in_progress","response":
{"id":"resp_0f2d32831fb00e2e0069d5f8e804808193b3379a80c7a3b172","object":"resp
onse","created_at":1775630568,"status":"in_progress","background":false,"compl
eted_at":null,"content_filters":null,"error":null,"frequency_penalty":0.0,"inc
omplete_details":null,"instructions":null,"max_output_tokens":null,"max_tool_c
alls":null,"model":"gpt-5","output":
[],"parallel_tool_calls":true,"presence_penalty":0.0,"previous_response_id":nu
ll,"prompt_cache_key":null,"prompt_cache_retention":null,"reasoning":
{"effort":"medium","summary":null},"safety_identifier":null,"service_tier":"au
to","store":true,"temperature":1.0,"text":{"format":
{"type":"text"},"verbosity":"medium"},"tool_choice":"auto","tools":
[],"top_logprobs":0,"top_p":1.0,"truncation":"disabled","usage":null,"user":nu
ll,"metadata":{}},"sequence_number":1}
data: {"type":"response.output_item.added","item":
{"id":"rs_0f2d32831fb00e2e0069d5f8e8e4d4819388a333a13963c26c","type":"reasonin
g","summary":[]},"output_index":0,"sequence_number":2}
data: {"type":"response.output_item.done","item":
{"id":"rs_0f2d32831fb00e2e0069d5f8e8e4d4819388a333a13963c26c","type":"reasonin
g","summary":[]},"output_index":0,"sequence_number":3}
data: {"type":"response.output_item.added","item":
{"id":"msg_0f2d32831fb00e2e0069d5f8eb771081939ac98a1af19dbfa6","type":"message
","status":"in_progress","content":
[],"role":"assistant"},"output_index":1,"sequence_number":4}
data:
{"type":"response.content_part.added","content_index":0,"item_id":"msg_0f2d328
31fb00e2e0069d5f8eb771081939ac98a1af19dbfa6","output_index":1,"part":
{"type":"output_text","annotations":[],"logprobs":
[],"text":""},"sequence_number":5}
data:
{"type":"response.output_text.delta","content_index":0,"delta":"I","item_id":"
msg_0f2d32831fb00e2e0069d5f8eb771081939ac98a1af19dbfa6","logprobs":
[],"obfuscation":"FLULWfR50RQt9xe","output_index":1,"sequence_number":6}
data: {"type":"response.output_text.delta","content_index":0,"delta":"
made","item_id":"msg_0f2d32831fb00e2e0069d5f8eb771081939ac98a1af19dbfa6","logp
robs":[],"obfuscation":"Aqz8i3eh080","output_index":1,"sequence_number":7}
data: {"type":"response.output_text.delta","content_index":0,"delta":"
a","item_id":"msg_0f2d32831fb00e2e0069d5f8eb771081939ac98a1af19dbfa6","logprob
s":[],"obfuscation":"gG89e4gtSwCR8X","output_index":1,"sequence_number":8}
data: {"type":"response.output_text.delta","content_index":0,"delta":"
pencil","item_id":"msg_0f2d32831fb00e2e0069d5f8eb771081939ac98a1af19dbfa6","lo
gprobs":[],"obfuscation":"kuiQ2eTZo","output_index":1,"sequence_number":9}
data: {"type":"response.output_text.delta","content_index":0,"delta":"
with","item_id":"msg_0f2d32831fb00e2e0069d5f8eb771081939ac98a1af19dbfa6","logp
robs":[],"obfuscation":"p6XFGSHfQYM","output_index":1,"sequence_number":10}
data: {"type":"response.output_text.delta","content_index":0,"delta":"
two","item_id":"msg_0f2d32831fb00e2e0069d5f8eb771081939ac98a1af19dbfa6","logpr
obs":[],"obfuscation":"VKPNZg4a8LVT","output_index":1,"sequence_number":11}
data: {"type":"response.output_text.delta","content_index":0,"delta":"
er","item_id":"msg_0f2d32831fb00e2e0069d5f8eb771081939ac98a1af19dbfa6","logpro
bs":[],"obfuscation":"hFwqXYfdYVgL6","output_index":1,"sequence_number":12}
data:
{"type":"response.output_text.delta","content_index":0,"delta":"asers","item_i
d":"msg_0f2d32831fb00e2e0069d5f8eb771081939ac98a1af19dbfa6","logprobs":
[],"obfuscation":"hJXekEK6KHi","output_index":1,"sequence_number":13}
data:
{"type":"response.output_text.delta","content_index":0,"delta":".","item_id":"
```

```
msg_0f2d32831fb00e2e0069d5f8eb771081939ac98a1af19dbfa6","logprobs":
[],"obfuscation":"3AZgVGpzDEHyrh0","output_index":1,"sequence_number":14}
data: {"type":"response.output_text.delta","content_index":0,"delta":"
It","item_id":"msg_0f2d32831fb00e2e0069d5f8eb771081939ac98a1af19dbfa6","logpro
bs":[],"obfuscation":"SgobRaEGNbRJv","output_index":1,"sequence_number":15}
data: {"type":"response.output_text.delta","content_index":0,"delta":"
was","item_id":"msg_0f2d32831fb00e2e0069d5f8eb771081939ac98a1af19dbfa6","logpr
obs":[],"obfuscation":"NJXJX0CC6d4y","output_index":1,"sequence_number":16}
data: {"type":"response.output_text.delta","content_index":0,"delta":"
pointless","item_id":"msg_0f2d32831fb00e2e0069d5f8eb771081939ac98a1af19dbfa6",
"logprobs":[],"obfuscation":"1c4UVp","output_index":1,"sequence_number":17}
data:
{"type":"response.output_text.delta","content_index":0,"delta":".","item_id":"
msg_0f2d32831fb00e2e0069d5f8eb771081939ac98a1af19dbfa6","logprobs":
[],"obfuscation":"nYlUlIwZ2i9vkBh","output_index":1,"sequence_number":18}
data:
{"type":"response.output_text.done","content_index":0,"item_id":"msg_0f2d32831
fb00e2e0069d5f8eb771081939ac98a1af19dbfa6","logprobs":
[],"output_index":1,"sequence_number":19,"text":"I made a pencil with two
erasers. It was pointless."}
data:
{"type":"response.content_part.done","content_index":0,"item_id":"msg_0f2d3283
1fb00e2e0069d5f8eb771081939ac98a1af19dbfa6","output_index":1,"part":
{"type":"output_text","annotations":[],"logprobs":[],"text":"I made a pencil
with two erasers. It was pointless."},"sequence_number":20}
data: {"type":"response.output_item.done","item":
{"id":"msg_0f2d32831fb00e2e0069d5f8eb771081939ac98a1af19dbfa6","type":"message
","status":"completed","content":[{"type":"output_text","annotations":
[],"logprobs":[],"text":"I made a pencil with two erasers. It was
pointless."}],"role":"assistant"},"output_index":1,"sequence_number":21}
data: {"type":"response.completed","response":
{"id":"resp_0f2d32831fb00e2e0069d5f8e804808193b3379a80c7a3b172","object":"resp
onse","created_at":1775630568,"status":"completed","background":false,"complet
ed_at":1775630571,"content_filters":
[{"blocked":false,"source_type":"prompt","content_filter_raw":
[],"content_filter_results":{"hate":
{"filtered":false,"severity":"safe"},"self_harm":
{"filtered":false,"severity":"safe"},"sexual":
{"filtered":false,"severity":"safe"},"violence":
{"filtered":false,"severity":"safe"}},"content_filter_offsets":
{"start_offset":1130,"end_offset":1151,"check_offset":0}},
{"blocked":false,"source_type":"completion","content_filter_raw":
[],"content_filter_results":{"self_harm":
{"filtered":false,"severity":"safe"},"sexual":
{"filtered":false,"severity":"safe"},"violence":
{"filtered":false,"severity":"safe"},"hate":
{"filtered":false,"severity":"safe"}},"content_filter_offsets":
{"start_offset":0,"end_offset":1163,"check_offset":0}}],"error":null,"frequenc
y_penalty":0.0,"incomplete_details":null,"instructions":null,"max_output_token
s":null,"max_tool_calls":null,"model":"gpt-5","output":
[{"id":"rs_0f2d32831fb00e2e0069d5f8e8e4d4819388a333a13963c26c","type":"reasoni
ng","summary":[]},
{"id":"msg_0f2d32831fb00e2e0069d5f8eb771081939ac98a1af19dbfa6","type":"message
","status":"completed","content":[{"type":"output_text","annotations":
[],"logprobs":[],"text":"I made a pencil with two erasers. It was
pointless."}],"role":"assistant"}],"parallel_tool_calls":true,"presence_penalt
y":0.0,"previous_response_id":null,"prompt_cache_key":null,"prompt_cache_reten
tion":null,"reasoning":
{"effort":"medium","summary":null},"safety_identifier":null,"service_tier":"au
to","store":true,"temperature":1.0,"text":{"format":
{"type":"text"},"verbosity":"medium"},"tool_choice":"auto","tools":
[],"top_logprobs":0,"top_p":1.0,"truncation":"disabled","usage":
{"input_tokens":12,"input_tokens_details":
{"cached_tokens":0},"output_tokens":286,"output_tokens_details":
{"reasoning_tokens":256},"total_tokens":298},"user":null,"metadata":
{}},"sequence_number":22}
```

Multi-turn Conversations

Sample Code

```
curl --location "$DEPLOYMENT_URL/responses" \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "model": "gpt-5",
  "input": "Count to 5 in Spanish."
}'
```

```
curl --location "$DEPLOYMENT_URL/responses" \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "model": "gpt-5",
  "input": "What did I just ask you to count to?",
  "previous_response_id":
"resp_0c7d5a12cbe135ee0069d602027e98819785c54c92cbb2c33c"
}'
```

Output Code


```
{
  "background": false,
  "completed_at": 1775632924,
  "content_filters": [
    {
      "blocked": false,
      "content_filter_offsets": {
        "check_offset": 0,
        "end_offset": 1303,
        "start_offset": 1267
      },
      "content_filter_raw": [],
      "content_filter_results": {
        "hate": {
          "filtered": false,
          "severity": "safe"
        },
        "self_harm": {
          "filtered": false,
          "severity": "safe"
        },
        "sexual": {
          "filtered": false,
          "severity": "safe"
        },
        "violence": {
          "filtered": false,
          "severity": "safe"
        }
      },
      "source_type": "prompt"
    },
    {
      "blocked": false,
      "content_filter_offsets": {
        "check_offset": 0,
        "end_offset": 405,
```

```
        "start_offset": 0
      },
      "content_filter_raw": [],
      "content_filter_results": {
        "hate": {
          "filtered": false,
          "severity": "safe"
        },
        "self_harm": {
          "filtered": false,
          "severity": "safe"
        },
        "sexual": {
          "filtered": false,
          "severity": "safe"
        },
        "violence": {
          "filtered": false,
          "severity": "safe"
        }
      },
      "source_type": "completion"
    }
  ],
  "created_at": 1775632922,
  "error": null,
  "frequency_penalty": 0.0,
  "id": "resp_0c7d5a12cbe135ee0069d6021a70048197b30eac95a9b0c028",
  "incomplete_details": null,
  "instructions": null,
  "max_output_tokens": null,
  "max_tool_calls": null,
  "metadata": {},
  "model": "gpt-5",
  "object": "response",
  "output": [
    {
      "id": "rs_0c7d5a12cbe135ee0069d6021ae4e08197aa623c16d9dd5c8f",
      "summary": [],
      "type": "reasoning"
    },
    {
      "content": [
        {
          "annotations": [],
          "logprobs": [],
          "text": "5",
          "type": "output_text"
        }
      ],
      "id": "msg_0c7d5a12cbe135ee0069d6021cb13c8197a301edad5a9c0dca",
      "role": "assistant",
      "status": "completed",
      "type": "message"
    }
  ],
  "parallel_tool_calls": true,
  "presence_penalty": 0.0,
  "previous_response_id":
"resp_0c7d5a12cbe135ee0069d602027e98819785c54c92cbb2c33c",
  "prompt_cache_key": null,
  "prompt_cache_retention": null,
  "reasoning": {
    "effort": "medium",
    "summary": null
  },
  "safety_identifier": null,
  "service_tier": "auto",
```

```
  "status": "completed",
  "store": true,
  "temperature": 1.0,
  "text": {
    "format": {
      "type": "text"
    },
    "verbosity": "medium"
  },
  "tool_choice": "auto",
  "tools": [],
  "top_logprobs": 0,
  "top_p": 1.0,
  "truncation": "disabled",
  "usage": {
    "input_tokens": 52,
    "input_tokens_details": {
      "cached_tokens": 0
    },
    "output_tokens": 120,
    "output_tokens_details": {
      "reasoning_tokens": 64
    },
    "total_tokens": 172
  },
  "user": null
}
```

Tool Calling

Sample Code

```
curl --location "$DEPLOYMENT_URL/responses" \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "model": "gpt-5",
  "input": "What is the weather in Berlin today?",
  "tools": [
    {
      "type": "function",
      "name": "get_weather",
      "description": "Retrieves current weather for a given location.",
      "parameters": {
        "type": "object",
        "properties": {
          "location": {
            "type": "string",
            "description": "City name, e.g. Berlin"
          },
          "unit": {
            "type": "string",
            "enum": ["celsius", "fahrenheit"],
            "description": "Temperature unit"
          }
        },
        "required": ["location"]
      }
    }
  ],
  "tool_choice": "auto"
```

```
}'
```

Output Code


```
{
  "background": false,
  "completed_at": 1775633033,
  "content_filters": [
    {
      "blocked": false,
      "content_filter_offsets": {
        "check_offset": 0,
        "end_offset": 66,
        "start_offset": 30
      },
      "content_filter_raw": [],
      "content_filter_results": {
        "hate": {
          "filtered": false,
          "severity": "safe"
        },
        "self_harm": {
          "filtered": false,
          "severity": "safe"
        },
        "sexual": {
          "filtered": false,
          "severity": "safe"
        },
        "violence": {
          "filtered": false,
          "severity": "safe"
        }
      },
      "source_type": "prompt"
    },
    {
      "blocked": false,
      "content_filter_offsets": {
        "check_offset": 0,
        "end_offset": 1964,
        "start_offset": 0
      },
      "content_filter_raw": [],
      "content_filter_results": {
        "hate": {
          "filtered": false,
          "severity": "safe"
        },
        "self_harm": {
          "filtered": false,
          "severity": "safe"
        },
        "sexual": {
          "filtered": false,
          "severity": "safe"
        },
        "violence": {
          "filtered": false,
          "severity": "safe"
        }
      },
      "source_type": "completion"
    }
  ],
  "created_at": 1775633030,
```

```
  "error": null,
  "frequency_penalty": 0,
  "id": "resp_047ee244b82be67b0069d60285fbf881968ffc2e359a601595",
  "incomplete_details": null,
  "instructions": null,
  "max_output_tokens": null,
  "max_tool_calls": null,
  "metadata": {},
  "model": "gpt-5",
  "object": "response",
  "output": [
    {
      "id": "rs_047ee244b82be67b0069d60286a7e48196bf5d878ac3101a4d",
      "summary": [],
      "type": "reasoning"
    },
    {
      "arguments": "{\"location\":\"Berlin\",\"unit\":\"celsius\"}",
      "call_id": "call_pcSA7YDBUkKraA3gPYC6BoCx",
      "id": "fc_047ee244b82be67b0069d60288a4e88196a01c79cb1955ce19",
      "name": "get_weather",
      "status": "completed",
      "type": "function_call"
    }
  ],
  "parallel_tool_calls": true,
  "presence_penalty": 0,
  "previous_response_id": null,
  "prompt_cache_key": null,
  "prompt_cache_retention": null,
  "reasoning": {
    "effort": "medium",
    "summary": null
  },
  "safety_identifier": null,
  "service_tier": "auto",
  "status": "completed",
  "store": true,
  "temperature": 1,
  "text": {
    "format": {
      "type": "text"
    },
    "verbosity": "medium"
  },
  "tool_choice": "auto",
  "tools": [
    {
      "description": "Retrieves current weather for a given location.",
      "name": "get_weather",
      "parameters": {
        "additionalProperties": false,
        "properties": {
          "location": {
            "description": "City name, e.g. Berlin",
            "type": "string"
          },
          "unit": {
            "description": "Temperature unit",
            "enum": [
              "celsius",
              "fahrenheit"
            ],
            "type": "string"
          }
        },
        "required": [
          "location",
```

```
          "unit"
        ],
        "type": "object"
      },
      "strict": true,
      "type": "function"
    }
  ],
  "top_logprobs": 0,
  "top_p": 1,
  "truncation": "disabled",
  "usage": {
    "input_tokens": 78,
    "input_tokens_details": {
      "cached_tokens": 0
    },
    "output_tokens": 175,
    "output_tokens_details": {
      "reasoning_tokens": 128
    },
    "total_tokens": 253
  },
  "user": null
}
```


Tuning Reasoning Effort

Sample Code

```
curl --location "$DEPLOYMENT_URL/responses" \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "model": "gpt-5",
  "input": "Solve: If a train travels 120 km in 1.5 hours, what is its
average speed?",
  "reasoning": {
    "effort": "high",
    "summary": "concise"
  }
}'
```

Output Code


```
{
  "background": false,
  "completed_at": 1775635760,
  "content_filters": [
    {
      "blocked": false,
      "content_filter_offsets": {
        "check_offset": 0,
        "end_offset": 1204,
        "start_offset": 1131
      },
      "content_filter_raw": [],
      "content_filter_results": {
        "hate": {
```

```
            "filtered": false,
            "severity": "safe"
          },
          "self_harm": {
            "filtered": false,
            "severity": "safe"
          },
          "sexual": {
            "filtered": false,
            "severity": "safe"
          },
          "violence": {
            "filtered": false,
            "severity": "safe"
          }
        },
        "source_type": "prompt"
      },
      {
        "blocked": false,
        "content_filter_offsets": {
          "check_offset": 0,
          "end_offset": 464,
          "start_offset": 0
        },
        "content_filter_raw": [],
        "content_filter_results": {
          "hate": {
            "filtered": false,
            "severity": "safe"
          },
          "self_harm": {
            "filtered": false,
            "severity": "safe"
          },
          "sexual": {
            "filtered": false,
            "severity": "safe"
          },
          "violence": {
            "filtered": false,
            "severity": "safe"
          }
        },
        "source_type": "completion"
      }
    ],
    "created_at": 1775635752,
    "error": null,
    "frequency_penalty": 0.0,
    "id": "resp_0fd09a736f9430c10069d60d28e08c8196b8a4960041311c66",
    "incomplete_details": null,
    "instructions": null,
    "max_output_tokens": null,
    "max_tool_calls": null,
    "metadata": {},
    "model": "gpt-5",
    "object": "response",
    "output": [
      {
        "id": "rs_0fd09a736f9430c10069d60d299a908196bb8d9631ee09094e",
        "summary": [
          {
            "text": "**Calculating average speed**\n\nI need to solve for
average speed using the formula: speed = distance/time. For 120 km over
1.5 hours, I find the speed is 80 km/h. Checking the math: 120 divided by
1.5 gives 80 too. The explanation is straightforward: average speed equals
distance divided by time, so that's 120 km divided by 1.5 hours. I could also
```

```
express it in meters per second, but I'll keep it simple, as the user didn't
request that.",
          "type": "summary_text"
        }
      ],
      "type": "reasoning"
    },
    {
      "content": [
        {
          "annotations": [],
          "logprobs": [],
          "text": "Average speed = distance ÷ time = 120 km ÷ 1.5 h = 80 km/
h.",
          "type": "output_text"
        }
      ],
      "id": "msg_0fd09a736f9430c10069d60d301c588196ae389bfac11f1621",
      "role": "assistant",
      "status": "completed",
      "type": "message"
    }
  ],
  "parallel_tool_calls": true,
  "presence_penalty": 0.0,
  "previous_response_id": null,
  "prompt_cache_key": null,
  "prompt_cache_retention": null,
  "reasoning": {
    "effort": "high",
    "summary": "concise"
  },
  "safety_identifier": null,
  "service_tier": "auto",
  "status": "completed",
  "store": true,
  "temperature": 1.0,
  "text": {
    "format": {
      "type": "text"
    },
    "verbosity": "medium"
  },
  "tool_choice": "auto",
  "tools": [],
  "top_logprobs": 0,
  "top_p": 1.0,
  "truncation": "disabled",
  "usage": {
    "input_tokens": 28,
    "input_tokens_details": {
      "cached_tokens": 0
    },
    "output_tokens": 162,
    "output_tokens_details": {
      "reasoning_tokens": 128
    },
    "total_tokens": 190
  },
  "user": null
}
```

Structured Output

Use `text.format` to constrain the model's output to a JSON schema. Set `strict: <true>` to enforce exact schema compliance

Sample Code

```
curl --location "$DEPLOYMENT_URL/responses" \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "model": "gpt-5",
  "input": "Extract the name and age from: John is 30 years old.",
  "text": {
    "format": {
      "type": "json_schema",
      "name": "person",
      "strict": true,
      "schema": {
        "type": "object",
        "properties": {
          "name": { "type": "string" },
          "age":  { "type": "integer" }
        },
        "required": ["name", "age"],
        "additionalProperties": false
      }
    }
  }
}'
```

Output Code

```
{
  "background": false,
  "completed_at": 1775660677,
  "content_filters": [
    {
      "blocked": false,
      "content_filter_offsets": {
        "check_offset": 0,
        "end_offset": 1321,
        "start_offset": 1269
      },
      "content_filter_raw": [],
      "content_filter_results": {
        "hate": {
          "filtered": false,
          "severity": "safe"
        },
        "self_harm": {
          "filtered": false,
          "severity": "safe"
        },
        "sexual": {
          "filtered": false,
          "severity": "safe"
        },
        "violence": {
          "filtered": false,
          "severity": "safe"
        }
```

```
      },
      "source_type": "prompt"
    },
    {
      "blocked": false,
      "content_filter_offsets": {
        "check_offset": 0,
        "end_offset": 1533,
        "start_offset": 0
      },
      "content_filter_raw": [],
      "content_filter_results": {
        "hate": {
          "filtered": false,
          "severity": "safe"
        },
        "self_harm": {
          "filtered": false,
          "severity": "safe"
        },
        "sexual": {
          "filtered": false,
          "severity": "safe"
        },
        "violence": {
          "filtered": false,
          "severity": "safe"
        }
      },
      "source_type": "completion"
    }
  ],
  "created_at": 1775660669,
  "error": null,
  "frequency_penalty": 0.0,
  "id": "resp_0261fdc22b6a21c50069d66e7d76ec8194bcf4bad9e6697593",
  "incomplete_details": null,
  "instructions": null,
  "max_output_tokens": null,
  "max_tool_calls": null,
  "metadata": {},
  "model": "gpt-5",
  "object": "response",
  "output": [
    {
      "id": "rs_0261fdc22b6a21c50069d66e7e05c08194b3f66d0adb40909d",
      "summary": [],
      "type": "reasoning"
    },
    {
      "content": [
        {
          "annotations": [],
          "logprobs": [],
          "text": "{\"name\":\"John\",\"age\":30}",
          "type": "output_text"
        }
      ],
      "id": "msg_0261fdc22b6a21c50069d66e8581b881948631e41a40256fa8",
      "role": "assistant",
      "status": "completed",
      "type": "message"
    }
  ],
  "parallel_tool_calls": true,
  "presence_penalty": 0.0,
  "previous_response_id": null,
  "prompt_cache_key": null,
```

```
  "prompt_cache_retention": null,
  "reasoning": {
    "effort": "medium",
    "summary": null
  },
  "safety_identifier": null,
  "service_tier": "auto",
  "status": "completed",
  "store": true,
  "temperature": 1.0,
  "text": {
    "format": {
      "description": null,
      "name": "person",
      "schema": {
        "additionalProperties": false,
        "properties": {
          "age": {
            "type": "integer"
          },
          "name": {
            "type": "string"
          }
        },
        "required": [
          "name",
          "age"
        ],
        "type": "object"
      },
      "strict": true,
      "type": "json_schema"
    },
    "verbosity": "medium"
  },
  "tool_choice": "auto",
  "tools": [],
  "top_logprobs": 0,
  "top_p": 1.0,
  "truncation": "disabled",
  "usage": {
    "input_tokens": 48,
    "input_tokens_details": {
      "cached_tokens": 0
    },
    "output_tokens": 354,
    "output_tokens_details": {
      "reasoning_tokens": 320
    },
    "total_tokens": 402
  },
  "user": null
}
```


Background Mode

Set `background: <true>` to run a long-running response asynchronously. Poll `GET /responses/{id}` to check status, or cancel with `POST /responses/{id}/cancel`.

Sample Code

```
curl --location "$DEPLOYMENT_URL/responses" \
```

```
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "model": "gpt-5",
  "input": "Write a 1 page report on AI trends.",
  "background": true
}'
```

Output Code


```
{
  "background": true,
  "completed_at": null,
  "content_filters": null,
  "created_at": 1775660765,
  "error": null,
  "frequency_penalty": 0.0,
  "id": "resp_00a26cdc24b0dd070069d66edd127c819598a74a3355ed43b4",
  "incomplete_details": null,
  "instructions": null,
  "max_output_tokens": null,
  "max_tool_calls": null,
  "metadata": {},
  "model": "gpt-5",
  "object": "response",
  "output": [],
  "parallel_tool_calls": true,
  "presence_penalty": 0.0,
  "previous_response_id": null,
  "prompt_cache_key": null,
  "prompt_cache_retention": null,
  "reasoning": {
    "effort": "medium",
    "summary": null
  },
  "safety_identifier": null,
  "service_tier": "auto",
  "status": "queued",
  "store": true,
  "temperature": 1.0,
  "text": {
    "format": {
      "type": "text"
    },
    "verbosity": "medium"
  },
  "tool_choice": "auto",
  "tools": [],
  "top_logprobs": 0,
  "top_p": 1.0,
  "truncation": "disabled",
  "usage": null,
  "user": null
}
```


Sample Code

```
curl "$DEPLOYMENT_URL/responses/<response_id>" \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header "Authorization: Bearer $AUTH_TOKEN"
```

Output Code


```
{
  "background": true,
  "completed_at": null,
  "content_filters": null,
  "created_at": 1775660765,
  "error": {
    "code": "server_error",
    "message": "An error occurred while processing your request. You can
retry your request, or contact us through an Azure support request at:
https://go.microsoft.com/fwlink/?linkid=2213926 if the error persists. Please
include the request ID wfr_019d6da1103b753ca4c0e3ed0bd8c380 in your message."
  },
  "frequency_penalty": 0.0,
  "id": "resp_00a26cdc24b0dd070069d66edd127c819598a74a3355ed43b4",
  "incomplete_details": null,
  "instructions": null,
  "max_output_tokens": null,
  "max_tool_calls": null,
  "metadata": {},
  "model": "gpt-5",
  "object": "response",
  "output": [],
  "parallel_tool_calls": true,
  "presence_penalty": 0.0,
  "previous_response_id": null,
  "prompt_cache_key": null,
  "prompt_cache_retention": null,
  "reasoning": {
    "effort": "medium",
    "summary": null
  },
  "safety_identifier": null,
  "service_tier": "auto",
  "status": "failed",
  "store": true,
  "temperature": 1.0,
  "text": {
    "format": {
      "type": "text"
    },
    "verbosity": "medium"
  },
  "tool_choice": "auto",
  "tools": [],
  "top_logprobs": 0,
  "top_p": 1.0,
  "truncation": "disabled",
  "usage": null,
  "user": null
}
```


Output Code


```
{
  "background": true,
  "completed_at": null,
  "content_filters": null,
  "created_at": 1775661040,
  "error": null,
  "frequency_penalty": 0,
  "id": "resp_00e6f5f17ced9f150069d66ff06ae88190b59be9ab2616d9d3",
  "incomplete_details": null,
  "instructions": null,
```

```
  "max_output_tokens": null,
  "max_tool_calls": null,
  "metadata": {},
  "model": "gpt-5",
  "object": "response",
  "output": [],
  "parallel_tool_calls": true,
  "presence_penalty": 0,
  "previous_response_id": null,
  "prompt_cache_key": null,
  "prompt_cache_retention": null,
  "reasoning": {
    "effort": "medium",
    "summary": null
  },
  "safety_identifier": null,
  "service_tier": "auto",
  "status": "in_progress",
  "store": true,
  "temperature": 1,
  "text": {
    "format": {
      "type": "text"
    },
    "verbosity": "medium"
  },
  "tool_choice": "auto",
  "tools": [],
  "top_logprobs": 0,
  "top_p": 1,
  "truncation": "disabled",
  "usage": null,
  "user": null
}
```

Output Code

```
{
  "background": true,
  "completed_at": 1775661083,
  "content_filters": [
    {
      "blocked": false,
      "content_filter_offsets": {
        "check_offset": 0,
        "end_offset": 1165,
        "start_offset": 1130
      },
      "content_filter_raw": [],
      "content_filter_results": {
        "hate": {
          "filtered": false,
          "severity": "safe"
        },
        "self_harm": {
          "filtered": false,
          "severity": "safe"
        },
        "sexual": {
          "filtered": false,
          "severity": "safe"
        },
        "violence": {
          "filtered": false,
          "severity": "safe"
        }
```

```
      },
      "source_type": "prompt"
    },
    {
      "blocked": false,
      "content_filter_offsets": {
        "check_offset": 0,
        "end_offset": 8975,
        "start_offset": 0
      },
      "content_filter_raw": [],
      "content_filter_results": {
        "hate": {
          "filtered": false,
          "severity": "safe"
        },
        "self_harm": {
          "filtered": false,
          "severity": "safe"
        },
        "sexual": {
          "filtered": false,
          "severity": "safe"
        },
        "violence": {
          "filtered": false,
          "severity": "safe"
        }
      },
      "source_type": "completion"
    }
  ],
  "created_at": 1775661040,
  "error": null,
  "frequency_penalty": 0.0,
  "id": "resp_00e6f5f17ced9f150069d66ff06ae88190b59be9ab2616d9d3",
  "incomplete_details": null,
  "instructions": null,
  "max_output_tokens": null,
  "max_tool_calls": null,
  "metadata": {},
  "model": "gpt-5",
  "object": "response",
  "output": [
    {
      "id": "rs_00e6f5f17ced9f150069d66ff199d08190a7b6ac8fe41f3e78",
      "summary": [],
      "type": "reasoning"
    },
    {
      "content": [
        {
          "annotations": [],
          "logprobs": [],
          "text": "AI Trends: A Brief Report\n\nSummary\nArtificial
intelligence is shifting from single-task models to integrated systems
that plan, call tools, and operate across text, images, audio, and
video. Organizations are moving from pilots to production, emphasizing
reliability, governance, and cost control. The focus is no longer just
“bigger models,” but right-sizing, data quality, and system design that
blends models with retrieval, tools, and human oversight.\n\nKey Trends\n\n-
From copilots to agents\n  - Evolution from assistive “copilots” to
agentic systems that break tasks into steps, call APIs, browse, write
code, and act.\n  - Tool use (function calling), planning, and memory
are becoming standard features, often orchestrated by workflow engines.\n
- Emphasis on guardrails, deterministic steps for critical actions, and
human-in-the-loop checkpoints.\n\n- Multimodal AI becomes default\n  - Models
that understand and generate across text, images, audio, and video are
```

```
increasingly practical.\n  - Use cases: knowledge search with charts and
images, call-center QA on transcripts, inspection and defect detection,
meeting summaries with slide context, and video analytics.\n  - OCR and
table understanding are integrated into general-purpose models, reducing
bespoke pipelines.\n\n- Smaller, specialized models rise\n  - “Small language
models” (SLMs) and distilled variants deliver adequate performance at
a fraction of cost and latency.\n  - Domain-tuned and instruction-tuned
models outperform larger general models on narrow tasks.\n  - On-device
and edge inference grows for privacy, latency, and offline use.\n\n- Data
and retrieval over sheer parameter count\n  - Retrieval-augmented generation
(RAG) is standard: grounding answers in structured and unstructured corpora
to cut hallucinations.\n  - Trend toward RAG 2.0: better chunking, hybrid
search (sparse + dense), graph-augmented retrieval, and citations with
confidence.\n  - Synthetic data generation is common for rare cases
and preference tuning, raising governance needs for provenance and bias
control.\n\n- Training and alignment advances\n  - Shift from RLHF alone
to a mix of methods: direct preference optimization (DPO), reinforcement
learning from AI feedback (RLAIF), and iterative self-play for tools
and code.\n  - Preference modeling increasingly task- and org-specific;
model behavior shaped by policy libraries and domain rules.\n\n- Inference
efficiency and architectures\n  - Mixture-of-experts (MoE), speculative
decoding, and KV-cache management reduce latency and cost.\n  - Quantization
and low-rank adaptation (LoRA) enable on-prem/edge deployments.\n  -
Throughput, context length, and structured output reliability matter more
than headline benchmarks.\n\n- Reliability, evaluation, and verifiability\n
- Production teams treat LLM systems like software: test suites, adversarial
evaluation, and continuous monitoring for drift.\n  - Techniques to reduce
hallucinations: retrieval-first prompting, constrained decoding, citations,
programmatic checks, and self-consistency voting.\n  - Model claims are
increasingly tied to evidence: provenance metadata, watermarking/signatures
for generated media, and audit trails.\n\n- Security and privacy\n  -
Prompt injection, data exfiltration, model poisoning, and jailbreaks are
now standard threat models.\n  - Countermeasures: input/output filters,
tool permissioning, sandboxing, rate limits, and red-teaming.\n  - Federated
learning and differential privacy remain relevant where regulated data
is involved.\n\n- Governance and regulation\n  - Organizations align with
emerging frameworks (e.g., NIST AI RMF) and sector-specific rules.\n
- System cards, model cards, incident response plans, and vendor risk
assessments are becoming procurement staples.\n  - Documentation, consent,
and IP considerations for training and outputs are front-and-center.\n\n-
Open vs. proprietary ecosystems\n  - Open models improve rapidly,
enabling customization and offline use; licensing and safety vary.\n  -
Proprietary frontier models still lead on complex reasoning and multimodal
fidelity, especially at long context.\n  - Many stacks adopt a hybrid:
open models for routine tasks, proprietary for high-stakes or complex
reasoning.\n\nImplications for Organizations\n\n- Start with the system, not
the model: combine retrieval, tools, and guardrails; measure end-to-end
outcomes.\n- Optimize for total cost of ownership: choose the smallest model
that meets quality; exploit quantization and caching.\n- Invest in data
pipelines: document quality, build feedback loops, and track provenance for
both real and synthetic data.\n- Treat safety and security as engineering
disciplines: red-team, monitor, and restrict tool permissions.\n- Plan for
change: abstract model providers behind interfaces; keep prompts, tools,
and evaluations portable.\n\nKey Metrics to Track\n\n- Quality: task-specific
pass rates, grounded citation rate, factuality and consistency.\n- Cost/
latency: cost per 1K/1M tokens, tokens/sec, P95/P99 latency.\n- Reliability:
hallucination rate, tool-call accuracy, eval drift over time.\n- Adoption:
task coverage, time-to-completion, user satisfaction, and ROI.",
          "type": "output_text"
        }
      ],
      "id": "msg_00e6f5f17ced9f150069d6700505e481909b3f2c01fc4e5411",
      "role": "assistant",
      "status": "completed",
      "type": "message"
    }
  ],
  "parallel_tool_calls": true,
```

```
  "presence_penalty": 0.0,
  "previous_response_id": null,
  "prompt_cache_key": null,
  "prompt_cache_retention": null,
  "reasoning": {
    "effort": "medium",
    "summary": null
  },
  "safety_identifier": null,
  "service_tier": "auto",
  "status": "completed",
  "store": true,
  "temperature": 1.0,
  "text": {
    "format": {
      "type": "text"
    },
    "verbosity": "medium"
  },
  "tool_choice": "auto",
  "tools": [],
  "top_logprobs": 0,
  "top_p": 1.0,
  "truncation": "disabled",
  "usage": {
    "input_tokens": 16,
    "input_tokens_details": {
      "cached_tokens": 0
    },
    "output_tokens": 1946,
    "output_tokens_details": {
      "reasoning_tokens": 896
    },
    "total_tokens": 1962
  },
  "user": null
}
```


Stateless (store: false)

> **Note**
>
> By default, the **`store`** parameter is set to `<false>`, and responses are not stored.
>
> Optionally, you can set **`store`** to `<true>` and responses could then be stored by Microsoft for up to 30 days. Saved requests can be deleted by sending a DELETE request to the endpoint `/v1/responses/<response_id>`.

Pass the full conversation history in `input` on each turn. Use include: `["reasoning.encrypted_content"]` to carry reasoning tokens across turns.

> **Sample Code**

```
curl --location "$DEPLOYMENT_URL/responses" \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "model": "gpt-5",
```

```
  "store": false,
  "input": [
    { "role": "user",      "content": "Count to 3 in Spanish." },
    { "role": "assistant", "content": "Uno, dos, tres." },
    { "role": "user",      "content": "Now count to 5." }
  ]
}'
```

Output Code


```
{
  "background": false,
  "completed_at": 1775661288,
  "content_filters": [
    {
      "blocked": false,
      "content_filter_offsets": {
        "check_offset": 0,
        "end_offset": 1253,
        "start_offset": 1238
      },
      "content_filter_raw": [],
      "content_filter_results": {
        "hate": {
          "filtered": false,
          "severity": "safe"
        },
        "self_harm": {
          "filtered": false,
          "severity": "safe"
        },
        "sexual": {
          "filtered": false,
          "severity": "safe"
        },
        "violence": {
          "filtered": false,
          "severity": "safe"
        }
      },
      "source_type": "prompt"
    },
    {
      "blocked": false,
      "content_filter_offsets": {
        "check_offset": 0,
        "end_offset": 732,
        "start_offset": 0
      },
      "content_filter_raw": [],
      "content_filter_results": {
        "hate": {
          "filtered": false,
          "severity": "safe"
        },
        "self_harm": {
          "filtered": false,
          "severity": "safe"
        },
        "sexual": {
          "filtered": false,
          "severity": "safe"
        },
        "violence": {
          "filtered": false,
          "severity": "safe"
```

```
        }
      },
      "source_type": "completion"
    }
  ],
  "created_at": 1775661284,
  "error": null,
  "frequency_penalty": 0,
  "id": "resp_083ce1293c8376680169d670e4bec481968f1ed8a8e235e7c2",
  "incomplete_details": null,
  "instructions": null,
  "max_output_tokens": null,
  "max_tool_calls": null,
  "metadata": {},
  "model": "gpt-5",
  "object": "response",
  "output": [
    {
      "id": "rs_083ce1293c8376680169d670e57b548196b9d623b0def49a52",
      "summary": [],
      "type": "reasoning"
    },
    {
      "content": [
        {
          "annotations": [],
          "logprobs": [],
          "text": "Uno, dos, tres, cuatro, cinco.",
          "type": "output_text"
        }
      ],
      "id": "msg_083ce1293c8376680169d670e7cb7c81969f7f0e7c85cc0433",
      "role": "assistant",
      "status": "completed",
      "type": "message"
    }
  ],
  "parallel_tool_calls": true,
  "presence_penalty": 0,
  "previous_response_id": null,
  "prompt_cache_key": null,
  "prompt_cache_retention": null,
  "reasoning": {
    "effort": "medium",
    "summary": null
  },
  "safety_identifier": null,
  "service_tier": "auto",
  "status": "completed",
  "store": false,
  "temperature": 1,
  "text": {
    "format": {
      "type": "text"
    },
    "verbosity": "medium"
  },
  "tool_choice": "auto",
  "tools": [],
  "top_logprobs": 0,
  "top_p": 1,
  "truncation": "disabled",
  "usage": {
    "input_tokens": 35,
    "input_tokens_details": {
      "cached_tokens": 0
    },
    "output_tokens": 185,
```

```
    "output_tokens_details": {
      "reasoning_tokens": 128
    },
    "total_tokens": 220
  },
  "user": null
}
```

Truncation

Set `truncation: "auto"` to automatically drop the oldest messages when the conversation exceeds the model's context limit. The default `"disabled"` returns an error instead.

Sample Code

```
curl --location "$DEPLOYMENT_URL/responses" \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "model": "gpt-5",
  "input": "Summarize our conversation so far.",
  "previous_response_id":
"resp_00e6f5f17ced9f150069d66ff06ae88190b59be9ab2616d9d3",
  "truncation": "auto"
}'
```

Output Code

```
{
  "background": false,
  "completed_at": 1775661392,
  "content_filters": [
    {
      "blocked": false,
      "content_filter_offsets": {
        "check_offset": 0,
        "end_offset": 6493,
        "start_offset": 6459
      },
      "content_filter_raw": [],
      "content_filter_results": {
        "hate": {
          "filtered": false,
          "severity": "safe"
        },
        "self_harm": {
          "filtered": false,
          "severity": "safe"
        },
        "sexual": {
          "filtered": false,
          "severity": "safe"
        },
        "violence": {
          "filtered": false,
          "severity": "safe"
        }
      },
```

```
      "source_type": "prompt"
    },
    {
      "blocked": false,
      "content_filter_offsets": {
        "check_offset": 0,
        "end_offset": 3224,
        "start_offset": 0
      },
      "content_filter_raw": [],
      "content_filter_results": {
        "hate": {
          "filtered": false,
          "severity": "safe"
        },
        "self_harm": {
          "filtered": false,
          "severity": "safe"
        },
        "sexual": {
          "filtered": false,
          "severity": "safe"
        },
        "violence": {
          "filtered": false,
          "severity": "safe"
        }
      },
      "source_type": "completion"
    }
  ],
  "created_at": 1775661382,
  "error": null,
  "frequency_penalty": 0.0,
  "id": "resp_00e6f5f17ced9f150069d6714554908190b834ab02e37bd4ab",
  "incomplete_details": null,
  "instructions": null,
  "max_output_tokens": null,
  "max_tool_calls": null,
  "metadata": {},
  "model": "gpt-5",
  "object": "response",
  "output": [
    {
      "id": "rs_00e6f5f17ced9f150069d67146af848190801dd7eca493f431",
      "summary": [],
      "type": "reasoning"
    },
    {
      "content": [
        {
          "annotations": [],
          "logprobs": [],
          "text": "Here's a brief recap:\n\n- You asked for a 1-page
report on AI trends.\n- I provided a report highlighting:\n  - Shift from
copilots to agentic systems with tool use, planning, and guardrails.\n  -
Multimodal AI (text, image, audio, video) becoming standard.\n  - Rise of
smaller/specialized and on-device models for cost, latency, and privacy.\n
- Emphasis on data quality and retrieval (RAG), including advanced retrieval
techniques.\n  - Training/alignment advances (e.g., DPO, RLAIF) and domain-
specific preference modeling.\n  - Inference efficiency (MoE, speculative
decoding, KV-cache, quantization, LoRA) and focus on throughput/context.\n
- Reliability and verifiability via testing, citations, constrained
decoding, and monitoring.\n  - Security/privacy threats and mitigations;
red-teaming and sandboxing.\n  - Governance/regulation practices and hybrid
open/proprietary model strategies.\n- Practical guidance included system-
first design, TCO optimization, strong data pipelines, safety/security as
```

```
engineering, and provider abstraction.\n- Suggested metrics: quality, cost/
latency, reliability, and adoption/ROI.",
          "type": "output_text"
        }
      ],
      "id": "msg_00e6f5f17ced9f150069d6714c8d908190ac535907666545f7",
      "role": "assistant",
      "status": "completed",
      "type": "message"
    }
  ],
  "parallel_tool_calls": true,
  "presence_penalty": 0.0,
  "previous_response_id":
"resp_00e6f5f17ced9f150069d66ff06ae88190b59be9ab2616d9d3",
  "prompt_cache_key": null,
  "prompt_cache_retention": null,
  "reasoning": {
    "effort": "medium",
    "summary": null
  },
  "safety_identifier": null,
  "service_tier": "auto",
  "status": "completed",
  "store": true,
  "temperature": 1.0,
  "text": {
    "format": {
      "type": "text"
    },
    "verbosity": "medium"
  },
  "tool_choice": "auto",
  "tools": [],
  "top_logprobs": 0,
  "top_p": 1.0,
  "truncation": "auto",
  "usage": {
    "input_tokens": 1065,
    "input_tokens_details": {
      "cached_tokens": 0
    },
    "output_tokens": 692,
    "output_tokens_details": {
      "reasoning_tokens": 384
    },
    "total_tokens": 1757
  },
  "user": null
}
```


Prompt Caching

Use `prompt_cache_key` to group similar requests and maximize cache reuse. Set prompt_cache_retention to control how long the cache entry is kept.

Sample Code

```
curl --location "$DEPLOYMENT_URL/responses" \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header 'Content-Type: application/json' \
```

```
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "model": "gpt-5",
  "input": "What is the capital of France?",
  "prompt_cache_key": "user_session_abc123"
}'
```

Output Code


```
{
  "id": "resp_cache123",
  "object": "response",
  "status": "completed",
  "model": "gpt-5",
  "prompt_cache_key": "user_session_abc123",
  "prompt_cache_retention": "24h",
  "output": [
    {
      "type": "message",
      "role": "assistant",
      "content": [
        {
          "type": "output_text",
          "text": "The capital of France is Paris."
        }
      ]
    }
  ],
  "usage": {
    "input_tokens": 10,
    "input_tokens_details": {
      "cached_tokens": 8
    },
    "output_tokens": 9,
    "output_tokens_details": {
      "reasoning_tokens": 0
    },
    "total_tokens": 19
  }
}
```


1.3.6.1.2 Lifecycle Management

Manages the lifecycle of a response created using the OpenAI Responses API. It allows you to retrieve a response by ID, cancel an in-progress response, or permanently delete a response for cleanup and control purposes.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $AUTH_TOKEN
AI-Resource-Group	The resource group used in the activation steps

Header	Value
$DEPLOYMENT_URL	The deployment URL for your generative AI model. For more information, see Create a Deployment [page 194]. Alternatively, you can replace the `$DEPLOYMENT_URL` placeholder in the curl command with your deployment URL.

Retrieve a Response by Id

Sample Code

```
curl "$DEPLOYMENT_URL/responses/<response_id> \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header "Authorization: Bearer $AUTH_TOKEN"
```

Output Code


```
{
  "background": false,
  "completed_at": 1775630571,
  "content_filters": [
    {
      "blocked": false,
      "content_filter_offsets": {
        "check_offset": 0,
        "end_offset": 1151,
        "start_offset": 1130
      },
      "content_filter_raw": [],
      "content_filter_results": {
        "hate": {
          "filtered": false,
          "severity": "safe"
        },
        "self_harm": {
          "filtered": false,
          "severity": "safe"
        },
        "sexual": {
          "filtered": false,
          "severity": "safe"
        },
        "violence": {
          "filtered": false,
          "severity": "safe"
        }
      },
      "source_type": "prompt"
    },
    {
      "blocked": false,
      "content_filter_offsets": {
        "check_offset": 0,
        "end_offset": 1163,
        "start_offset": 0
      },
      "content_filter_raw": [],
      "content_filter_results": {
```

```
        "hate": {
          "filtered": false,
          "severity": "safe"
        },
        "self_harm": {
          "filtered": false,
          "severity": "safe"
        },
        "sexual": {
          "filtered": false,
          "severity": "safe"
        },
        "violence": {
          "filtered": false,
          "severity": "safe"
        }
      },
      "source_type": "completion"
    }
  ],
  "created_at": 1775630568,
  "error": null,
  "frequency_penalty": 0.0,
  "id": "resp_0f2d32831fb00e2e0069d5f8e804808193b3379a80c7a3b172",
  "incomplete_details": null,
  "instructions": null,
  "max_output_tokens": null,
  "max_tool_calls": null,
  "metadata": {},
  "model": "gpt-5",
  "object": "response",
  "output": [
    {
      "id": "rs_0f2d32831fb00e2e0069d5f8e8e4d4819388a333a13963c26c",
      "summary": [],
      "type": "reasoning"
    },
    {
      "content": [
        {
          "annotations": [],
          "logprobs": [],
          "text": "I made a pencil with two erasers. It was pointless.",
          "type": "output_text"
        }
      ],
      "id": "msg_0f2d32831fb00e2e0069d5f8eb771081939ac98a1af19dbfa6",
      "role": "assistant",
      "status": "completed",
      "type": "message"
    }
  ],
  "parallel_tool_calls": true,
  "presence_penalty": 0.0,
  "previous_response_id": null,
  "prompt_cache_key": null,
  "prompt_cache_retention": null,
  "reasoning": {
    "effort": "medium",
    "summary": null
  },
  "safety_identifier": null,
  "service_tier": "auto",
  "status": "completed",
  "store": true,
  "temperature": 1.0,
  "text": {
    "format": {
```

```
      "type": "text"
    },
    "verbosity": "medium"
  },
  "tool_choice": "auto",
  "tools": [],
  "top_logprobs": 0,
  "top_p": 1.0,
  "truncation": "disabled",
  "usage": {
    "input_tokens": 12,
    "input_tokens_details": {
      "cached_tokens": 0
    },
    "output_tokens": 286,
    "output_tokens_details": {
      "reasoning_tokens": 256
    },
    "total_tokens": 298
  },
  "user": null
}
```


Cancel a Response by Id

Sample Code

```
curl --request POST --location "$DEPLOYMENT_URL/responses/<response_id>/
cancel" \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header "Authorization: Bearer $AUTH_TOKEN"
```

Output Code


```
{
  "error": {
    "code": null,
    "message": "Cannot cancel a synchronous response.",
    "param": null,
    "type": "invalid_request_error"
  }
}
```


Delete a Response by Id

Sample Code

```
curl --request DELETE "$DEPLOYMENT_URL/responses/<response_id>" \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header "Authorization: Bearer $AUTH_TOKEN"
```

Output Code

```
{
  "deleted": true,
  "id": "resp_0f2d32831fb00e2e0069d5f8e804808193b3379a80c7a3b172",
  "object": "response.deleted"
}
```

1.3.7 Perplexity

Context

You can access LLMs through the `foundation-models` and `orchestration` scenarios.

To use a model through the `orchestration` scenario, using the harmonized API, see Orchestration [page 210].

For more information about available models, including conversion rates for tokens, rate limits, and deprecation dates, see SAP Note 3437766.

Accessing Models Through Orchestration

Orchestration offers a harmonized API that allows you to use different models without changing the client code. You likely have an orchestration deployment running in your default resource group. If you want to access models through orchestration in a different resource group, create an orchestration deployment for your chosen resource group. For more information, see .Create a Deployment for Orchestration [page 703].

Access to orchestration of generative AI models is provided under the global AI scenario `orchestration`, which is managed by SAP AI Core.

To access generative AI models using orchestration, add the following information to the `model` module of your orchestration workflow:

- The name of your chosen model
- The version name of your chosen version. If no model version is listed, it is not applicable.

For more information about supported models and associated costs, see SAP Note 3505347

For more information about orchestration workflows, see Orchestration Workflow V2 [page 382].

Accessing Models through the `foundation-models` Scenario

You can access foundation models by creating a deployment for the model that you want to use. To do so, you need an auth token from your SAP AI Core instance. For more information, see Get an Auth Token [page 6] and Create a Deployment [page 194].

To create your deployment, you'll need the following information:

- The scenario: `foundation-models`
- The `executableId`: `perplexity-ai`
- The name of your chosen model
- The version name of your chosen version. If no model version is listed, it is not applicable.

To use a specific version of a model or to upgrade model versions manually, specify the version of your model deployment. To upgrade automatically, specify the model version as `latest`. For more information, see Model Lifecycle [page 194]. If a model version isn't listed, the model isn't applicable.

Perplexity models are remote models.

After you've created a deployment for your model, you can consume the model using prompts. To access the model, you need your deployment ID, which you can set as an environment variable.

Check that you've set the following headers:

Header	Value
Authorization	Bearer $AUTH_TOKEN
AI-Resource-Group	The resource group used in the activation steps
$DEPLOYMENT_URL	The deployment URL for your generative AI model. For more information, see Create a Deployment [page 194]. Alternatively, you can replace the `$DEPLOYMENT_URL` placeholder in the curl command with your deployment URL.

Consumption

The following examples show how you can consume various generative AI models using curl. For more information about prompts, see the tutorial Prompt LLMs in the Generative AI Hub in SAP AI Core & Launchpad.

> → Tip
>
> If you use a Windows device, use Windows PowerShell, and replace `curl` with `curl.exe`.

For more information about supported parameters, see Supported Parameters [page 199].

Completions

sonar | sonar-pro

Sample Code

```
curl --location '$DEPLOYMENT_URL/chat/completions' \
--header 'AI-Resource-Group: default' \
--header 'Content-Type: application/json' \
--heder "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "model": "<modelName>",
    "stream": false,
    "max_tokens": 20,
    "messages": [
      {
        "role": "system",
        "content": "You are a powerful business AI assistant."
      },
      {
        "role": "user",
        "content": "Who is the president of the USA?"
      }
    ]
}'
```

Removing a Model

If you want to remove a model, delete its deployment. For more information, see Delete a Single Deployment

1.3.8 Prior Labs

Tabular foundation models from Prior Labs generate predictions from business data containing numerical, categorical, and textual information without requiring extensive data preparation or traditional model training. You can use these models to solve classification and regression problems across business functions.

The generative AI hub in SAP AI Core provides access to the TabPFN-3.5 family of tabular foundation models from Prior Labs, part of SAP. The family currently includes TabPFN-3.5 Plus, a model designed to solve prediction problems using tabular data across industries and business functions. This model helps you to incorporate accurate predictions into business processes while reducing the effort required for data preparation and model training.

Traditional machine learning approaches often assume that rows are independent, features are clean and well structured, and sufficient data is available for training. Business data, however, rarely meets all these assumptions. For example, an invoice can combine monetary values with customer identifiers, while a purchase order can contain supplier information, material descriptions, and requested delivery dates. This combination of numerical, categorical, textual, and other business data can provide valuable signals for predicting outcomes such as payment delays, forecasting cash flow, and supporting operational decisions.

Extracting these signals with conventional machine learning can require significant data preparation, feature engineering, and model training. TabPFN-3.5 is designed to reduce this effort, particularly for challenging

tabular datasets. It can work with text-rich tables, categorical features with many distinct values, and datasets containing large numbers of input features, without requiring extensive cleaning or transformation before the data can be used for prediction.

Business Use Cases

The TabPFN-3.5 family of models supports a wide range of prediction scenarios across business functions. These include:

- **Demand and inventory planning:** Estimate product demand to support purchasing decisions and optimize supply chain operations.
- **Finance and cash-flow management:** Predict which invoices are likely to be paid late to improve cash-flow forecasting and planning.
- **Supplier reliability:** Predict delays in the delivery of purchase orders to identify potential risks and mitigate production disruptions.
- **Production and quality management:** Identify production batches that are at risk of failing quality control checks.
- **Workforce planning:** Estimate the time required to complete maintenance activities and improve scheduling efficiency.
- **Sales forecasting:** Predict which prospects are most likely to convert and prioritize outreach activities.
- **Customer retention:** Identify customers at risk of churn and proactively engage them through support and retention programs.
- **Spend classification:** Predict purchasing categories from line-item data to improve spend analysis and reporting.

Prediction Tasks and Request Formats

The `/predict` endpoint supports classification and regression prediction tasks.

- Classification: Predicts a category value. A maximum of 160 classes is supported.
- Regression: Predicts a numeric value.

The `/predict` endpoint accepts the following request formats:

- `application/json`: Includes datasets directly in the JSON request body.
- `multipart/form-data`: Uploads datasets as `CSV` or Apache Parquet files. A `request` text field contains the request configuration.

Request Fields

The prediction request contains the following required fields.

Field	Required	Description
`x_train`	Yes	Training features. One row per labeled example.
`y_train`	Yes	Training target. Must contain exactly one column aligned row-for-row with `x_train`.
`x_test`	Yes	Rows for prediction. Must contain the same columns as `x_train`.
`task_config`	Yes	Task type and model configuration.

Task Configuration Fields

Field	Default	Description
`task_config.task`	Required	Prediction task type. Supported values are `classification` and `regression`.
`task_config.tabpfn_config`	{}	Model configuration settings.
`task_config.predict_params`	{}	Prediction output settings.

Model Configuration Options

The `tabpfn_config` object provides optional settings for model inference.

Configuration	Type	Default	Description
`n_estimators`	Integer	8	Number of ensemble members used for prediction
`softmax_temperature`	Float	Model default	Controls the confidence of predicted probability distributions
`average_before_softmax`	Boolean	Model default	Specifies whether ensemble outputs are averaged before or after softmax
`balance_probabilities`	Boolean	Model default	Rebalances class probabilities for classification tasks
`categorical_features_indices`	List of integers	Auto-detected	Defines columns that are to be treated as categorical features
`random_state`	Integer	Unset	Seed value for reproducible predictions

Configuration	Type	Default	Description
`inference_precision`	String	Model default	Numeric precision used during inference
`fit_mode`	Enum	fit_preprocessors	Preprocessing and caching strategy
`memory_saving_mode`	Boolean	Automatic	Reduces memory consumption for large datasets
`inference_config`	Object	Unset	Advanced inference settings

Classification Output Options

Output Type	Returns
`probas`	Per-class probabilities for each test row
`preds`	Predicted class label for each test row
`top_k`	Most probable classes ranked by likelihood; requires the `top_k` parameter

Regression Output Options

Output Type	Returns
`mean`	Predicted mean value
`median`	Predicted median value
`mode`	Predicted mode value
`quantiles`	Values for requested quantiles
`main`	Point estimates and quantile values
`full`	Prediction statistics and predictive distributions

Additional Parameters

- `quantiles`: List of values between 0 and 1 used with `quantiles` output
- `top_k`: Number of classes returned when using `top_k`

Response Fields

Field	Description
`prediction`	Prediction results
`class_labels`	Sorted list of class labels observed during training (available only for classification tasks)
`metadata`	Execution details
`usage`	Dataset statistics and ensemble information
`model_id`	Model identifier, if available

Prediction Result Formats

Task and Output Type	Shape
Classification / `probas`	List of class probability values for each prediction row
Classification / `preds`	List of predicted labels
Regression / `mean`, `median`, `mode`	List of numeric values
Regression / `quantiles`	List of quantile values
Regression / `full`	Object containing prediction statistics and distributions

> **Note**
>
> Non-finite values, such as positive or negative infinity, are serialized as `null`.

Metadata Fields

Field	Description
`task`	Processed prediction task
`test_set_num_rows`	Number of rows submitted for prediction
`test_set_num_cols`	Number of feature columns after preprocessing
`tabpfn_config`	Resolved model configuration, including default values applied during execution

1.3.8.1 TabPFN-3.5 Plus

Deploy the TabPFN-3.5 Plus model and use example requests to perform classification and regression tasks on tabular data.

The TabPFN-3.5 Plus model lets you solve prediction problems using tabular data across industries and business functions. The model helps you to incorporate accurate predictions into business processes while reducing the effort required for data preparation and model training.

Prerequisites

Before using TabPFN-3.5 in the generative AI hub, complete the onboarding steps required for SAP AI Core.

- You've completed the tasks described in Initial Setup.
- You're using the **extended** service plan, which is required for the generative AI hub.

Data Limits

TabPFN-3.5 Plus applies the following limits to prediction requests.

Limit Type	Supported Limit
Input context rows (with known outcomes)	1,000,000
Input columns	2,000
Context and prediction table size	200 million cells
Classification categories	160

Prediction requests support up to 1,000,000 prediction rows. Lower limits can apply depending on the size of the context dataset, the number of feature columns, and the size of the prediction output.

If a request contains 1,000,000 context rows, the maximum number of prediction rows is 250,000 before other limits are considered.

Responses are limited to 32 million output values. For example, when returning probabilities for all 160 classification categories, the maximum supported number of prediction rows is 200,000.

Full regression distribution outputs support a maximum of 400 prediction rows per request.

> **Note**
>
> The maximum number of supported prediction rows can vary based on the context dataset size, feature count, selected output type, and response size.

Deployment Information

You can access foundation models by creating a deployment for the model that you want to use. To do so, you need an auth token from your SAP AI Core instance. For more information, see Get an Auth Token [page 6] and Create a Deployment [page 194].

To create your deployment, you need the following information:

- Scenario: `foundation-models`
- `executableId`: `prior-labs`
- Model name: `tabpfn-3.5-plus`
- Model version: `2609`

> **Note**
>
> Make sure that the model is available in the generative AI hub before creating a deployment. For more information, see Supported Models .

To use a specific version of a model or to upgrade model versions manually, specify the version of your model deployment. To upgrade automatically, specify the model version as `latest`. For more information, see Model Lifecycle [page 194]. If a model version isn't listed, the model isn't applicable.

After you've created a deployment for your model, you can consume the model using prompts. To access the model, you need your deployment ID, which you can set as an environment variable.

Check that you've set the following headers:

Header	Value
Authorization	Bearer $AUTH_TOKEN
AI-Resource-Group	The resource group used in the activation steps
$DEPLOYMENT_URL	The deployment URL for your generative AI model. For more information, see Create a Deployment [page 194]. Alternatively, you can replace the `$DEPLOYMENT_URL` placeholder in the curl command with your deployment URL.

Examples

The following examples show how to submit prediction requests by using the `/predict` endpoint.

Example 1: Classification Request

The following example submits a classification request and returns the predicted class label.

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
```

```
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'Content-Type: application/json' \
  --data '{
  "task_config": {
    "task": "classification",
    "tabpfn_config": {
      "n_estimators": 8,
      "softmax_temperature": 0.9,
      "average_before_softmax": false,
      "balance_probabilities": true,
      "categorical_features_indices": [0, 2],
      "random_state": 0,
      "inference_precision": "auto",
      "fit_mode": "fit_preprocessors",
      "memory_saving_mode": false
    },
    "predict_params": {
      "output_type": "preds"
    }
  },
  "x_train": {
    "PRODUCT": ["Office Chair", "Server Rack"],
    "PRICE": [150.8, 2200],
    "ORDERDATE": ["10-1-2025", "12-1-2025"]
  },
  "y_train": {
    "COSTCENTER": ["Office Furniture", "Data Infrastructure"]
  },
  "x_test": {
    "PRODUCT": ["Couch"],
    "PRICE": [999.99],
    "ORDERDATE": ["10-2-2025"]
  }
}'
```

Example 2: Regression Request

The following example submits a regression request and returns the predicted mean value.

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'Content-Type: application/json' \
  --data '{
  "task_config": {
    "task": "regression",
    "predict_params": {
      "output_type": "mean"
    }
  },
  "x_train": [
    [5.0, 2.0],
    [2.0, 1.5],
    [3.2, 0.7],
    [0.8, 2.8]
  ],
  "y_train": [1.0, 1.8, 1.6, 1.8],
  "x_test": [
    [1.5, 1.9],
    [2.4, 0.8],
    [3.6, 4.5]
```

```
  ]
}'
```

Example 3: Regression Request with Advanced Configuration

The following example returns quantile predictions and configures additional inference options.

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'Content-Type: application/json' \
  --data '{
  "task_config": {
    "task": "regression",
    "predict_params": {
      "output_type": "quantiles",
      "quantiles": [0.1, 0.5, 0.9]
    },
    "tabpfn_config": {
      "n_estimators": 4,
      "random_state": 42,
      "memory_saving_mode": true
    }
  },
  "x_train": [
    [5.0, 2.0],
    [2.0, 1.5],
    [3.2, 0.7],
    [0.8, 2.8]
  ],
  "y_train": [1.0, 1.8, 1.6, 1.8],
  "x_test": [
    [1.5, 1.9],
    [2.4, 0.8]
  ]
}'
```

Example 4: Multipart Request

The following example uploads training and prediction datasets as CSV files and provides the prediction configuration in the `request` field.

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  -F 'x_train=@x_train.csv' \
  -F 'y_train=@y_train.csv' \
  -F 'x_test=@x_test.csv' \
  -F 'request={"task_config":{"task":"classification","predict_params":
{"output_type":"preds"}}}'
```

Example 5: Classification Response

The following example shows a successful classification response returned by the `/predict` endpoint.


```
{
  "prediction": [[0.93, 0.05, 0.02], [0.08, 0.81, 0.11]],
  "class_labels": ["Data Infrastructure", "Office Furniture"],
  "metadata": {
    "task": "classification",
    "test_set_num_rows": 1,
    "test_set_num_cols": 11,
    "tabpfn_config": {
      "n_estimators": 8,
      "average_before_softmax": false,
      "balance_probabilities": true,
      "categorical_features_indices": [0, 2],
      "fit_mode": "fit_preprocessors",
      "inference_precision": "auto",
      "memory_saving_mode": false,
      "random_state": 0,
      "softmax_temperature": 0.9
    }
  },
  "usage": {
    "X_train": {
      "num_rows": 4,
      "num_cols": 3,
      "num_cells": 12
    },
    "y_train": {
      "num_rows": 4,
      "num_cols": 1,
      "num_cells": 4
    },
    "X_test": {
      "num_rows": 2,
      "num_cols": 3,
      "num_cells": 6
    },
    "num_cells": 22,
    "num_predictions": 2,
    "n_estimators": 8
  },
  "model_id": null
}
```


1.3.9 SAP

1.3.9.1 SAP-RPT

1.3.9.1.1 SAP-RPT-1

SAP-RPT-1 is a relational pretrained transformer for use on relational and structured data. It's developed and maintained by SAP.

Relational Foundation Models (RFMs) are large-scale machine learning models designed to understand, process, and do predictions on tabular and relational data.

SAP-RPT-1 is a table-native model aiming to achieve highest prediction quality and lowest error rates for predictions on tabular business data. It's pretrained, and doesn't need additional training or fine-tuning steps.

RPT-1 solves predictive tasks such as classification and regression out-of-the-box without requiring any training or fine-tuning via in-context learning. Due to its table-native architecture, prediction quality on enterprise data is typically very high, ahead of state-of-the-art narrow AI models and LLMs employed for such tasks.

You can get predictions directly in one API call from provided context examples.

Key Features

Key features include:

- **Pretrained Model:** Removes the need to collect training datasets, manage compute infrastructure, or wait weeks for model training cycles. Deploy immediately with production-ready accuracy.
- **In-Context Learning:** Accepts representative context examples during inference time to return instant predictions without the need for setup or customization for your specific use cases.
- **Fast Iterations and Adaptation:** Adapts to the context data provided in your requests, without the need for additional training steps or deployments.
- **Grounded to Business:** Produces predictions that are grounded in the provided context examples, resulting in a lower risk of prediction errors or hallucinations.

Tabular In-Context Learning

SAP-RPT-1 produces predictions through tabular in-context learning.

Prompts are given in table form. They contain context examples and fields marked for prediction with a placeholder value.

Example

The following prompt shows a table where cost centers are associated with ordered products. A placeholder value (`[PREDICT]`) has been used to request a prediction for the `COSTCENTER` of the couch. SAP-RPT-1 returns proposals for the prediction request. Multiple requests can be included in one table prompt.

PRODUCT	PRICE	ORDERDATE	ID	COSTCENTER
Couch	999.99	28-11-2025	35	`[PREDICT]`
Office Chair	150.8	02-11-2025	44	Office Furniture
Server Rack	2200.00	01-11-2025	104	Data Infrastructure

SAP RPT Models

SAP-RPT-1 has the following versions:

- `sap-rpt-1-small`, which is recommended when:
  - Prediction scenarios are of medium complexity.
  - Low latency and high-prediction throughput are the primary objective.
- `sap-rpt-1-large`, which is recommended when:
  - Prediction scenarios are complex.
  - Best prediction quality and lowest error rates are the primary objective.

Model Specifications

Value	`sap-rpt-1-small`	`sap-rpt-1-large`
Max. Context Length	2048 rows	65536 rows
Max. Columns	100 columns	256 columns
Number of Target Classes (Classification)	256*	1024*
Recommended Context Length	500–2000 rows+	4000–8000 rows+

Both model versions allow up to 128 simultaneous prediction rows and up to 10 simultaneous prediction columns in each prediction request. Supported task types are `classification` and `regression`.

* The number of target classes is a recommendation for best prediction quality, not a hard limit. Depending on the specific use case, models can produce high-quality predictions for larger numbers of target classes.

+ Generic recommendations on context length are best practice values based on internal tests, balancing prediction quality, model runtime, and costs. Optimal settings vary across use cases and data sets, so customers or implementation partners must conduct their own testing with production data to achieve the best results for their specific scenario.

Workflow

You can access the models through the `foundation-models` scenario and the `aicore-sap` executable ID.

To use a model through the `foundation-models` scenario, you must create a configuration and then a deployment. You need the `executableId`: `aicore-sap` and `modelName`: `sap-rpt-1-small` or `sap-rpt-1-large`. You will also need a model version, or you can use `latest` to use the latest version. For more information, see Create a Deployment [page 194].

You consume the model through API calls. For more information, see Example Payloads for Inferencing: sap-rpt-1 [page 124].

Costs

For an estimate of projected costs, use the For an estimate of projected costs, use the SAP AI Core cost calculator, and select your preferred model. For more information, see SAP AI Core Cost Calculator for Generative AI.

1.3.9.1.1.1 Example Payloads for Inferencing: `sap-rpt-1`

> → Tip
>
> If you use a Windows device, use Windows PowerShell, and replace `curl` with `curl.exe`.

Check that you've set the following headers:

Header	Value
Authorization	Bearer $AUTH_TOKEN
AI-Resource-Group	The resource group used in the activation steps
$DEPLOYMENT_URL	The deployment URL for your generative AI model. For more information, see Create a Deployment [page 194]. Alternatively, you can replace the `$DEPLOYMENT_URL` placeholder in the curl command with your deployment URL.

The `executableId` is `aicore-sap`.

Best Practices

To achieve the best possible outputs, ensure that your requests comply with the following best practices:

- Use clean data with no or minimum missing values.
- Use descriptive column names (under 100 characters).
- Limit descriptions in text columns to 500 characters.
- Keep data types consistent within each column.
- Consider removing unnecessary columns from the prediction task.

> ⓘ Note
>
> The accuracy and fairness of the model's predictions depend on the quality and balance of the data provided. Any bias present in the input data can result in biased or skewed outputs. SAP isn't responsible or liable for any bias, inaccuracy, or unintended outcome arising from user-provided data or its characteristics.

Request Payloads

All JSON payloads must be UTF-8 encoded.

`sap-rpt-1-small` and `sap-rpt-1-large` accept tabular data requests in the following formats:

- Data as JSON organized by rows
- Data as JSON organized by columns
- Data in Apache Parquet file format

You can make requests using the following methods:

Request with Data by Rows

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'content-type: application/json' \
  --data '{
    "prediction_config": {
        "target_columns": [
            {
                "name": "COSTCENTER",
                "prediction_placeholder": "[PREDICT]",
                "task_type": "classification",
                "top_k": 1
            }
        ]
    },
    "index_column": "ID",
    "rows": [
        {
            "PRODUCT": "Couch",
            "PRICE": 999.99,
            "ORDERDATE": "28-11-2025",
            "ID": "35",
            "COSTCENTER": "[PREDICT]"
        },
        {
            "PRODUCT": "Office Chair",
            "PRICE": 150.8,
            "ORDERDATE": "02-11-2025",
            "ID": "44",
            "COSTCENTER": "Office Furniture"
        },
        {
            "PRODUCT": "Server Rack",
            "PRICE": 2200.00,
            "ORDERDATE": "01-11-2025",
            "ID": "104",
            "COSTCENTER": "Data Infrastructure"
        }
    ],
    "data_schema": {
        "PRODUCT": {
            "dtype": "string"
        },
        "PRICE": {
            "dtype": "numeric"
        },
        "ORDERDATE": {
            "dtype": "date"
        },
```

```
        "ID": {
            "dtype": "string"
        },
        "COSTCENTER": {
            "dtype": "string"
        }
    }
}'
```

Request with Data by Columns

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'Content-Type: application/json' \
  --data '{
  "prediction_config": {
    "target_columns": [
      {
        "name": "COSTCENTER",
        "prediction_placeholder": "[PREDICT]",
        "task_type": "classification",
        "top_k": 1
      }
    ]
  },
  "index_column": "ID",
  "columns": {
      "PRODUCT": ["Couch", "Office Chair", "Server Rack"],
      "PRICE": [999.99, 150.8, 2200.00],
      "ORDERDATE": ["28-11-2025", "02-11-2025", "01-11-2025"],
      "ID": ["35", "44", "104"],
      "COSTCENTER": ["[PREDICT]", "Office Furniture", "Data Infrastructure"]
  },
  "data_schema": {
      "PRODUCT": {
          "dtype": "string"
      },
      "PRICE": {
          "dtype": "numeric"
      },
      "ORDERDATE": {
          "dtype": "date"
      },
      "ID": {
          "dtype": "string"
      },
      "COSTCENTER": {
          "dtype": "string"
      }
  }
}'
```

Request with Parquet File

Parquet files can be uploaded using multipart form-data.

The request must include the following form fields:

- `file`: The parquet file to be uploaded (`Content-Type: application/vnd.apache.parquet`)
- `prediction_config`: A JSON string containing the prediction configuration
- `index_column`: The name of the column used to identify the row
- `parse_data_types`: A boolean that specifies whether to parse data types automatically (default: `true`)

The values for the `index_column` and `parse_data_types` fields are optional but recommended. If you don't specify values for these fields, the system uses default values.

The request must use `Content-Type: multipart/form-data` with an appropriate boundary string. The parquet file is read and processed similarly to row-wise and column-wise JSON inputs.

```
POST $DEPLOYMENT_URL/predict_parquet
  --header "Content-Type: multipart/form-data; boundary=----
WebKitFormBoundary7MA4YWxkTrZu0gW" \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header "ai-resource-group: default"
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="file"; filename="test_data.parquet"
Content-Type: application/vnd.apache.parquet
< ./test_data.parquet
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="prediction_config"
{"target_columns": [{"name": "salary","prediction_placeholder": "[PREDICT]"}]}
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="index_column"
name
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="parse_data_types"
true
------WebKitFormBoundary7MA4YWxkTrZu0gW--
```

Parameters

Parameter Description for `/predict` (JSON input)

Name	Type	Description
`prediction_config`	object	The configuration object specifying which columns to predict
`prediction_config.target_columns`	array	An array of target-column configurations for prediction
`prediction_config.target_columns[].name`	string	The name of the column to predict
`prediction_config.target_columns[].prediction_placeholder`	string	The value used as placeholder in rows where prediction is needed

Name	Type	Description
`prediction_config.target_columns[].task_type`	string	(Optional) The type of prediction task Possible values: `classification` or `regression` If omitted, a regression task is assumed if at least one value in the input data is formatted like a float, (i.e. `"1.0"`), otherwise classification is performed.
`prediction_config.target_columns[].top_k`	integer	(Optional) default is 1. Only for task type "classification". Controls how many labels to predict for each column in each row.
`index_column`	string	(Optional) The name of the column used to identify the row This column isn't used as an input feature for the model, and is returned in the response objects.
`rows`	array	An array of objects representing table rows (both context and query rows) The keys of the objects are column names, for example `[{"product": "hammer", "price": 42.99, "creation_date": "2025-01-30"}, ...]`
`columns`	object	An object mapping from column name to array of column values. You can pass rows only or columns only, for example `{"product": ["hammer", "nail", ...], "price": [42.99, 1.98, ...], "creation_date": ["2025-01-30", "2025-01-31", ...]}`
`data_schema`	object	(Optional) A schema definition for data types of all columns, for example `{"columnA": {"dtype": "string"}, "columnB": {"dtype": "numeric"}, ...}`

Name	Type	Description
`data_schema.*.dtype`	string	A data type specification. Possible values: `string`, `numeric`, or `date`
`parse_data_types`	Boolean	Relevant when `data_schema` isn't passed Whether or not to parse data types, such as interpreting strings as numbers or dates Default value: `true`

The `data_schema parameter` and `task_type` parameters are optional, but are recommended to obtain the best prediction results.

If `data_schema` is omitted, data types are inferred automatically.

If `task_type` is omitted, the model attempts to determine whether `classification` or `regression` is the intended task type.

Parameter description for `/predict_parquet` (Parquet file input as multipart/form-data)

Name	Type	Description
`prediction_config`	object	The configuration object specifying which columns to predict
`prediction_config.target_columns`	array	The array of target-column configurations for prediction
`prediction_config.target_columns[].name`	string	The name of the column to predict
`prediction_config.target_columns[].prediction_placeholder`	string	The value used as placeholder in rows where prediction is needed
`prediction_config.target_columns[].top_k`	integer	(Optional) default is 1. Only for task type "classification". Controls how many labels to predict for each column in each row.

Name	Type	Description
`prediction_config.target_columns[].task_type`	string	(Optional) The type of prediction task Possible values: `classification` or `regression` If omitted, the model tries to automatically determine the task type that the user intends.
`index_column`	string	(Optional) The name of the column used to identify the row This column isn't used as an input feature for the model, and is returned in the response objects.
`file`	application/	A parquet file

Response Payload

The objects in `predictions` are sequence-aligned with the order of the query rows in the request JSON.

If the `index_column` parameter has been passed, the contents of this column are also included in the response.

To align the response to the source data, you can sequence-align the arrays, or compare the contents of the index column where included.

Name	Type	Description
`id`	string	The generated UUID4 for this request (useful for logging)
`status`	object	The status message object indicating success of the prediction request
`status.code`	integer	The numeric status code. For more information, see Status Codes and Messages [page 131].
`status.message`	string	The status message text
`predictions[].*[].prediction`	string or float	The predicted value for this column
`predictions[].*[].confidence`	float or null	Confidence value for this classification prediction (null for regression)

Name	Type	Description
predictions[].*[].confidence_interval	array or null	Confidence interval for this regression prediction as array of two floats `[lower_bound, upper_bound]` or `null` for classification
`metadata.num_rows`	integer	The total number of input rows
`metadata.num_columns`	integer	The total number of input columns
`metadata.num_predictions`	integer	The number of table cells containing the specified placeholder values summed over all target columns
metadata.num_query_rows	integer	The number of query rows for which a prediction was made

> **Note**
>
> Error responses can contain an additional array called `detail`, following pydantic format. For more information, see Status Codes and Messages [page 131].

Status Codes and Messages

A status code is returned to indicate success, a warning, or an error. Status codes can be any of the following:

Status Codes and Messages

Code	Message	Description
0	OK	Everything worked as expected.
1	Warning: At least one target column had invalid values in some context rows (expected float-like values for regression). These rows were dropped. Double-check your data; prediction accuracy may be impacted.	Sub optimal data/accuracy warning still returns predictions.
2	Invalid input	The input was invalid: It could contain, for example. • Malformed JSON • Missing properties • Incorrectly typed properties • Maximum restrictions exceeded
3	Internal server error	An unexpected error occurred.

Successful Prediction (200)

If the parameter `index_column` is specified in the request JSON, the following apply:

- Each prediction object in the array contains an index column. In the following examples, the index column is called `ID`.
- The column referenced as the `index_column` isn't input to the model and serves as index information in the response JSON.

Output Code

```
{
    "id": "c334f854-0d70-4c79-bd73-9ac581fd8cda",
    "status": {
        "code": 0,
        "message": "ok"
    },
    "predictions": [
        {
            "COSTCENTER": [
                {
                    "prediction": "Office Furniture",
                    "confidence": 0.96,
                    "confidence_interval": null
                }
            ],
            "ID": "35"
        }
    ],
    "metadata": {
        "num_columns": 5,
        "num_rows": 2,
        "num_predictions": 1,
        "num_query_rows": 1
    }
}
```

Successful Prediction (200) with Warning

This example shows an output where the request to perform a regression contains either empty strings or null values in the regression target column.

```
{
    "id": "bf9018c4-f903-4035-a433-7058cb289447",
    "status": {
        "code": 1,
        "message": "Warning: At least one target column had invalid values in
some context rows (expected float-like values for regression). These rows were
dropped. Double-check your data, prediction accuracy may be impacted."
    },
    "predictions": [
        {
            "salary": [
                {
                    "prediction": 52451.3984375,
                    "confidence": null,
                    "confidence_interval": [50000.0, 58500.0]
                }
            ]
        }
    ],
    "metadata": {
        "num_columns": 4,
        "num_rows": 2,
        "num_predictions": 1,
        "num_query_rows": 1
    }
```

```
}
```

Bad Request or Unprocessable Content (400 or 422)

```
{
    "status": {
        "code": 2,
        "message": "Invalid input"
    },
    "detail": [
        {
            "loc": [],
            "msg": "Too many query rows provided. Maximum is 512.",
            "type": "value_error"
        }
    ]
}
```

```
{
    "status": {
        "code": 2,
        "message": "Invalid input"
    },
    "detail": [
        {
            "loc": [
                "prediction_config",
                "target_columns",
                0,
                "prediction_placeholder"
            ],
            "msg": "Field required",
            "type": "missing"
        }
    ]
}
```

Internal Server Error (500)

Output Code

```
  "status": {
     "code": 3,
     "message": "Internal server error"
 }
```

1.3.9.1.2 SAP-RPT-1.5

SAP-RPT-1.5 is a relational pre-trained transformer for prediction tasks on structured and relational data. It's developed and maintained by SAP.

For general information about the SAP-RPT model family, including key features, tabular in-context learning, deployment instructions, API usage, and response formats, see SAP-RPT-1 [page 121] overview.

SAP RPT Models

For more information about `SAP-RPT-1.5`, see SAP 1.5 Model Card

`SAP-RPT-1.5` is available in the following model versions:

- `sap-rpt-1.5` supports medium-complexity prediction scenarios. It is optimized for low latency and high prediction throughput.
- `sap-rpt-1.5-large` supports complex prediction scenarios. It is optimized for high prediction quality and low error rates.

> **Note**
>
> SAP-RPT-1.5 predicts multiple target columns together. As a result, predictions can differ when you predict multiple target columns in a single request instead of predicting each target column separately.

Model Specifications

Value	sap-rpt-1.5	sap-rpt-1.5-large
Max. Context Length	2048 rows	65536 rows
Max. Columns	100 columns	256 columns
Number of Target Classes (Classification)	256*	1024*
Recommended Context Length	500-2000 rows+	4000-8000 rows+

Both model versions support up to 512 rows and 10 columns per prediction request. Supported task types are `classification` and `regression`.

* Number of target classes does not represent a hard limit. SAP recommends these values to achieve optimal prediction quality. Depending on the use case, larger numbers of target classes can also produce high-quality predictions.

+ Recommended context lengths are based on SAP experimentation and represent best-practice values that balance prediction quality, runtime, and cost. Depending on the application scenario and business requirements, different settings can provide better results. SAP recommends testing with production data to identify the optimal configuration.

Explainability

`SAP-RPT-1.5` supports explainability capabilities that help you understand how the model generates predictions.

For information about explainability, request parameters, response fields, and example requests and responses, see SAP RPT SAP RPT 1.5 Explainability [page 135].

1.3.9.1.2.1 SAP RPT 1.5 Explainability

`SAP-RPT-1.5` supports an explainability feature that provides insight into how the model generates predictions.

Enable explainability by adding an `explanations` configuration to `prediction_config` in the request.

> **Note**
>
> Column scores and relevant context rows are derived from model internals and don't indicate causality. Results can be less accurate for small datasets. For correlated features, the model can distribute scores across multiple feature columns instead of assigning a high score to a single column.

Request Parameters

Name	Type	Description
`prediction_config.explanations`	object	Optional. Configuration used to request explanations. The default value is `null`.
`prediction_config.explanations.top_column_scores`	integer	Optional. Number of top column scores to return. The default value is `0`. A value of `0` disables explainability. The maximum value is `20`.
`prediction_config.explanations.top_relevant_context_rows`	integer	Optional. Number of most relevant context rows to return for each query row. The maximum value is `20`.

Input Format Guidance

You can provide input tables as column-wise JSON (`columns`), row-wise JSON (`rows`), or a Parquet file.

For JSON requests, SAP recommends using `columns` because each column name is sent only once, reducing payload size. Use `rows` when readability is more important than payload size. The service processes requests in the same way regardless of whether you use `columns` or `rows`.

For JSON requests, consider providing the `data_schema` parameter to specify data types. Alternatively, use a Parquet file that contains the appropriate data types.

Example Request

Add the `explanations` object to `prediction_config`.

```
"prediction_config": {
    "target_columns": [
        {
            "name": "COSTCENTER",
            "prediction_placeholder": "[PREDICT]",
            "task_type": "classification",
            "top_k": 1
        }
    ],
    "explanations": {
        "top_column_scores": 3,
        "top_relevant_context_rows": 2
    }
}
```

Response Fields

Name	Type	Description
`explanations`	object or null	Explanation data containing column scores and relevant context rows. Present only when explainability is enabled.
`explanations.top_column_scores`	array or null	Array of objects, one for each query row, that maps feature columns to attention-based importance scores. Higher scores indicate a stronger contribution to the prediction. The service returns only the requested top-k columns.
`explanations.top_relevant_context_rows`	array or null	Contains the indices of the most relevant context rows for each query row, sorted by relevance. Indices are zero-based positions in the original input dataset. The service returns only the requested top-k rows.

Interpreting top_relevant_context_rows

The `top_relevant_context_rows` field identifies the rows from the input dataset that contributed most to a prediction. To use these values correctly, keep the same input table that you submitted in the prediction request.

Each value returned in `top_relevant_context_rows` represents a zero-based row position in the original input table. This behavior is the same for row-wise JSON input, column-wise JSON input, and Parquet input.

For JSON requests, column-wise input is typically more compact because each column name is specified only once.

The following example shows a column-wise JSON request:

```
{
  "columns": {
    "id": [1, 2, 3, 4],
    "product": ["Laptop", "Mouse", "Keyboard", "Monitor"],
    "category": ["Electronics", "Accessories", "Accessories", "?"]
  }
}
```

The prediction response can return the following explanation:

```
{
  "explanations": {
    "top_relevant_context_rows": [
      [1, 2]
    ]
  }
}
```

In this example, query row `0` refers to the first prediction row in the response. The values `1` and `2` identify the most relevant rows from the original input table.

The following example retrieves the corresponding rows from the original request payload:

```
input_columns = payload["columns"]
for row_idx in result["explanations"]["top_relevant_context_rows"][0]:
    row = {
        column_name: w_idx]
        for column_name, values in input_columns.items()
    }
```

```
    print(f"API row {row_idx}: {row}")
```

The output can look similar to the following:

```
API row 1: {'id': 2, 'product': 'Mouse', 'category': 'Accessories'}
API row 2: {'id': 3, 'product': 'Keyboard', 'category': 'Accessories'}
```

Query row indexes and API input row positions represent different values. Query row index `0` refers to the first item in the response explanation array, while API row positions `1` and `2` refer to rows in the original input table. These values are row positions and not the values from the `id` column.

When using Parquet input, preserve the row order from the uploaded table and use the returned row positions against that same table.

Example Response with Explanations


```
{
  "id": "c334f854-0d70-4c79-bd73-9ac581fd8cda",
  "status": {
    "code": 0,
    "message": "ok"
  },
  "predictions": [
    {
      "COSTCENTER": [
        {
          "prediction": "Office Furniture",
          "confidence": 0.96,
          "confidence_interval": null
        }
      ],
      "ID": "35"
    }
  ],
  "explanations": {
    "top_column_scores": [
      {
        "PRODUCT": 0.523,
        "PRICE": 0.234,
        "ORDERDATE": 0.121
      }
    ],
    "top_relevant_context_rows": [
      [1, 2]
    ]
  },
  "metadata": {
    "num_columns": 5,
    "num_rows": 2,
    "num_predictions": 1,
    "num_query_rows": 1
  }
}
```

1.3.9.1.3 SAP-RPT-1.6

SAP-RPT-1.6 is a relational pre-trained transformer for prediction tasks on structured and relational data. It supports classification and regression tasks and is developed and maintained by SAP.

For general information about the SAP-RPT model family, including key features, tabular in-context learning, deployment instructions, API usage, and response formats, see SAP-RPT-1 [page 121] overview.

SAP-RPT Models

For more information about `SAP-RPT-1.6`, see SAP 1.6 Model Card

`SAP-RPT-1.6` is available in the following model versions:

- `sap-rpt-1.6` supports medium-complexity prediction scenarios. It is optimized for low latency and high prediction throughput.
- `sap-rpt-1.6-large` supports complex prediction scenarios. It is optimized for high prediction quality and low error rates.

> **Note**
>
> SAP-RPT-1.6 predicts multiple target columns together. As a result, predictions can differ when you predict multiple target columns in a single request instead of predicting each target column separately.

Model Specifications

Value	sap-rpt-1.6	sap-rpt-1.6-large
Max. Context Length	2048 rows	65536 rows
Max. Query Length	512 rows	512 rows
Max. Columns	100 columns	256 columns
Max. Target Columns	10	10
Number of Target Classes (Classification)	250[1]	5000[1]
Recommended Context Length	500-2000 rows[2]	4000-36000 rows[2]

Supported task types are `classification` and `regression`.

[1] Number of target classes does not represent a hard limit. SAP recommends these values to achieve optimal prediction quality. Depending on the use case, larger numbers of target classes can also produce high-quality predictions.

[2] Recommended context lengths are based on SAP experimentation and represent best-practice values that balance prediction quality, runtime, and cost. Depending on the application scenario and business requirements, different settings can provide better results. SAP recommends testing with production data to identify the optimal configuration.

Context Mode

`sap-rpt-1.6-large` supports the optional `context_mode` parameter in the `prediction_config` object.

The `context_mode` parameter enables higher prediction accuracy when a large number of context rows is available.

> **Note**
>
> Setting `context_mode` to `deep` increases prediction latency. SAP recommends using deep context mode only when more than 8000 context rows are available and additional prediction accuracy is required.

SAP recommends evaluating `sap-rpt-1.6` first. If additional accuracy is required, use `sap-rpt-1.6-large` with the default context mode before enabling deep context mode.

Request Parameters

Name	Type	Description
`prediction_config.context_mode`	string	Optional. The default value is `default`. Use `default` for the best balance between accuracy and latency. Use `deep` only with `sap-rpt-1.6-large` when more than 8000 context rows are available and higher accuracy is required.

Example Request

```
"prediction_config": {
    "target_columns": [
        {
            "name": "COSTCENTER",
            "prediction_placeholder": "[PREDICT]",
            "task_type": "classification",
            "top_k": 1
        }
    ],
    "context_mode": "deep"
}
```

Explainability

SAP-RPT-1.6 supports explainability capabilities that help you understand how the model generates predictions.

For information about explainability, request parameters, response fields, and example requests and responses, see SAP RPT SAP RPT 1.6 Explainability [page 140].

1.3.9.1.3.1 SAP RPT 1.6 Explainability

`SAP-RPT-1.6` supports an explainability feature that provides insight into how the model generates predictions.

Enable explainability by adding an `explanations` configuration to `prediction_config` in the request.

> **Note**
>
> Column scores and relevant context rows are derived from model internals and do not indicate causality. Results can be less accurate for small datasets. For strongly correlated features, the model can distribute scores across multiple feature columns instead of assigning a high score to a single column.

Request Parameters

Name	Type	Description
`prediction_config.explanations`	object	Optional. Configuration used to request explanations. The default value is `null`.
`prediction_config.explanations.top_column_scores`	integer	Optional. Number of top column scores to return. The default value is `0`. A value of `0` disables explainability. The maximum value is `20`.
`prediction_config.explanations.top_relevant_context_rows`	integer	Optional. Number of most relevant context rows to return for each query row. The maximum value is `20`.

Input Format Guidance

You can provide input tables as column-wise JSON (`columns`), row-wise JSON (`rows`), or a Parquet file.

For JSON requests, SAP recommends using `columns` because each column name is sent only once, reducing payload size. Use `rows` when readability is more important than payload size. The service processes requests in the same way regardless of whether you use `columns` or `rows`.

For JSON requests, consider providing the `data_schema` parameter to specify data types. Alternatively, use a Parquet file that contains the appropriate data types.

Example Request

Add the `explanations` object to `prediction_config`.

```
"prediction_config": {
    "target_columns": [
        {
            "name": "COSTCENTER",
            "prediction_placeholder": "[PREDICT]",
            "task_type": "classification",
            "top_k": 1
        }
    ],
    "explanations": {
        "top_column_scores": 3,
        "top_relevant_context_rows": 2
    }
}
```

Response Fields

Name	Type	Description
`explanations`	object or null	Explanation data containing column scores and relevant context rows. Present only when explainability is enabled.
`explanations.top_column_scores`	array or null	Array of objects, one for each query row, that maps feature columns to attention-based importance scores. Higher scores indicate a stronger contribution to the prediction. The service returns only the requested top-k columns.
`explanations.top_relevant_context_rows`	array or null	Contains the indices of the most relevant context rows for each query row, sorted by relevance. Indices are zero-based positions in the original input dataset. The service returns only the requested top-k rows.

Interpreting top_relevant_context_rows

The `top_relevant_context_rows` field identifies the rows from the input dataset that contributed most to a prediction. To use these values correctly, keep the same input table that you submitted in the prediction request.

Each value returned in `top_relevant_context_rows` represents a zero-based row position in the original input table. This behavior is the same for row-wise JSON input, column-wise JSON input, and Parquet input.

For JSON requests, column-wise input is typically more compact because each column name is specified only once.

The following example shows a column-wise JSON request:

```
{
  "columns": {
    "id": [1, 2, 3, 4],
    "product": ["Laptop", "Mouse", "Keyboard", "Monitor"],
    "category": ["Electronics", "Accessories", "Accessories", "?"]
  }
}
```

The prediction response can return the following explanation:

```
{
  "explanations": {
    "top_relevant_context_rows": [
      [1, 2]
    ]
  }
}
```

In this example, query row `0` refers to the first prediction row in the response. The values `1` and `2` identify the most relevant rows from the original input table.

The following example retrieves the corresponding rows from the original request payload:

```
input_columns = payload["columns"]
for row_idx in result["explanations"]["top_relevant_context_rows"][0]:
    row = {
        column_name: w_idx]
        for column_name, values in input_columns.items()
    }
```

```
    print(f"API row {row_idx}: {row}")
```

The output can look similar to the following:

```
API row 1: {'id': 2, 'product': 'Mouse', 'category': 'Accessories'}
API row 2: {'id': 3, 'product': 'Keyboard', 'category': 'Accessories'}
```

Query row indexes and API input row positions represent different values. Query row index `0` refers to the first item in the response explanation array, while API row positions `1` and `2` refer to rows in the original input table. These values are row positions and not the values from the `id` column.

When using Parquet input, preserve the row order from the uploaded table and use the returned row positions against that same table.

For a complete runnable example that prints row mappings and explanation details, see the sample file sap rpt samples in the SAP RPT samples repository.

Example Response with Explanations

```
{
  "id": "c334f854-0d70-4c79-bd73-9ac581fd8cda",
  "status": {
    "code": 0,
    "message": "ok"
  },
  "predictions": [
    {
      "COSTCENTER": [
        {
          "prediction": "Office Furniture",
          "confidence": 0.96,
          "confidence_interval": null
        }
      ],
      "ID": "35"
    }
  ],
  "explanations": {
    "top_column_scores": [
      {
        "PRODUCT": 0.523,
        "PRICE": 0.234,
        "ORDERDATE": 0.121
      }
    ],
    "top_relevant_context_rows": [
      [1, 2]
    ]
  },
  "metadata": {
    "num_columns": 5,
    "num_rows": 2,
    "num_predictions": 1,
    "num_query_rows": 1
  }
}
```

1.3.9.1.4 SAP-RPT-TS

SAP-RPT-TS is a pretrained time series foundation model from SAP that provides high-quality forecasts for business time series data such as demand, sales, and operational metrics without requiring training or fine-tuning.

SAP-RPT-TS extends the RPT model family from static tabular data to temporal and sequential data. The model is designed to deliver highly accurate forecasts with low prediction error rates for enterprise time series use cases.

SAP-RPT-TS uses in-context learning to generate forecasts directly from historical time series data. By providing recent observations and optionally related series or covariates, you can obtain forecasts without training or fine-tuning a model.

The model returns probabilistic forecasts as predictive distributions rather than single point estimates. This enables uncertainty quantification and supports more informed business decisions.

You can generate forecasts directly through a single API call by providing the required historical context.

Key Features

- **Pretrained Model:** Use the model immediately without collecting training datasets, managing training infrastructure, or running model training cycles.
- **Fast and Lightweight:** The compact architecture delivers low-latency forecasts and runs efficiently on standard CPU infrastructure, reducing deployment costs and simplifying scaling across multiple time series.
- **In-Context Learning:** Provide recent historical observations during inference and receive forecasts instantly without creating or maintaining individual forecasting models.
- **Flexible Data Structures:** Support univariate and multivariate targets, multiple target variables, past covariates, and known future covariates to improve forecast accuracy.
- **Probabilistic and Grounded:** Generate calibrated prediction intervals in addition to forecast values, helping reduce the risk of overprediction or underprediction.

SAP-RPT-TS Models

SAP-RPT-TS is currently available in the following model version:

- `sap-rpt-ts and version : 1`

Model Specifications

Specification	sap-rpt-ts-1
Maximum Context Length	8192 rows
Maximum Horizon Steps	1024
Recommended Feature Columns	1–128 columns

Supported input structures include univariate targets, multivariate targets, past covariates, and future covariates.

Workflow

You can access SAP-RPT-TS through the `foundation-models` scenario using the executable ID `aicore-sap`.

To use the model, create a configuration and then create a deployment using the following values:

- **Executable ID:** `aicore-sap`
- **Model Name:** `sap-rpt-ts and version : 1`
- **Model Version:** Specify a version or use `latest`.

For more information about deployments, see Create a Deployment [page 194].

After deployment, consume the model through API calls. For sample requests and responses, see Inference Request Overview [page 145].

Costs

To estimate projected costs, use the SAP AI Core cost calculator and select the SAP-RPT-TS model. For more information, see SAP AI Core Cost Calculator for Generative AI.

Best Practices

To achieve the best forecasting results, follow these recommendations.

Input Data

- Provide as many historical observations as possible. Short input histories can reduce forecast quality.
- A single time series can contain up to 8192 observations. Longer series are automatically truncated to the most recent 8192 observations.
- Set the forecast horizon according to the required prediction length. Values greater than 1024 are automatically truncated to 1024.
- Use regularly spaced timestamps whenever possible. The model performs best when observations are evenly spaced.
- A limited number of missing timestamps is supported. Internally generated timestamps must not exceed 20% of the supplied timestamps.
- Provide past and future covariates whenever available. Forecast quality improves significantly when relevant explanatory variables are supplied.
- For optimal inference performance, keep the combined number of targets and covariates below 128 per series.

Choose an Appropriate Point Forecast Method

Choose the `point_forecast` method that best matches your loss function and data characteristics:

- `point_forecast: "median"` returns q=0.5 directly (MAE-optimal and robust to heavy-tailed distributions). This is the default option and is recommended for general-purpose forecasting of smooth, continuous demand.
- `point_forecast: "mean"` approximates the predictive mean by using trapezoidal integration over the `quantile_levels` grid (MSE/RMSE-optimal). It is recommended for intermittent demand with many zero values. Use a dense `quantile_levels` grid to improve mean estimation accuracy.
- `point_forecast: "hierarchical_forecast"` is recommended when the series follows a known aggregation hierarchy, such as country → city → street, and forecasts at all levels must remain consistent. Define the hierarchy using `hierarchical_labels` and specify aggregation levels in `hierarchical_levels`, ordered from the coarsest to the finest level. Use `null` to represent the grand total. The service forecasts each level and reconciles the results using MinT-WLS so that parent forecasts equal the sum of their child forecasts at each horizon step.

Control Forecast Output

- Set `output_quantile: true` when you need the full predictive distribution (shape `[n_series][n_variates][K][horizon]`, where K=21) for custom prediction intervals, client-side point estimators, or pinball-loss evaluation. The `output_quantile` option always returns the fixed 21-quantile grid regardless of `quantile_levels`. The `quantile_levels` option controls only the `selected_quantile_result` field.
- Use `quantile_levels` when you require only specific quantiles. For example, `[0.1, 0.5, 0.9]` returns only those quantiles in `selected_quantile_result` instead of the full 21-quantile output. For `mean`, the specified quantiles also define the integration grid used to calculate the point forecast.
- Use `clip_min` and `clip_max` to constrain the point forecast. These settings are applied only to `result` and do not affect `confidence_interval_output` or quantile outputs. As a result, a clipped forecast can fall outside the reported confidence interval.
- Enable `cross_learning: true` when the time series in a request are related and can benefit from shared information. Leave this option set to `false` (default) when the series are unrelated and should be forecast independently.
- Enable `enhanced_forecast: true` when forecast accuracy is the highest priority. This option applies additional inference techniques that can improve forecast quality at the cost of increased inference time. Input time-point consumption is billed at a flat 4× rate when enabled, regardless of history length or forecast horizon. Forecast horizon points are not multiplied.

> **Note**
>
> The accuracy and fairness of the model's forecasts depend on the quality, length, regularity, and balance of the data provided. Any bias present in the input data can result in biased or skewed outputs. SAP is not responsible for any bias, inaccuracy, or unintended outcome arising from user-provided data.

1.3.9.1.4.1 Inference Request Overview

The Time Series Foundation Model provides a single inference endpoint at `$DEPLOYMENT_URL/predict`. The endpoint generates multi-step probabilistic forecasts for one or more time series and supports optional

covariates, quantile forecasts, confidence intervals, cross-series learning, hierarchical forecasting, and forecast post-processing.

> → Tip
>
> If you use a Windows device, use Windows PowerShell and replace `curl` with `curl.exe`.

Prerequisites

Ensure that you have created and deployed a time series model. For more information, see Create a Deployment.

Ensure that you've set the following headers:

Header	Value
Authorization	Bearer $AUTH_TOKEN
AI-Resource-Group	The resource group used in the activation steps.

Make sure you have set the following variable:

- **$AI_API_URL**: The base URL of your SAP AI Core environment. You can set this URL as an environment variable.

Use the deployment URL as the value of `$DEPLOYMENT_URL` or replace the `$DEPLOYMENT_URL` placeholder in the example requests with your deployment URL.

Request Payload

All JSON payloads must be UTF-8 encoded.

The `/predict` endpoint accepts a JSON body with three top-level fields: `horizon`, `data` (a list of series), and an optional `options` block.

Forecast Request

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'content-type: application/json' \
  --data '{
    "horizon": 3,
    "data": [
        {
            "timestamp": ["<timestamp 1>", "<timestamp 2>", "..."],
            "target": [
                ["<variate 1 timepoint 1>", "<variate 1 timepoint 2>", "..."]
            ],
            "past_covariates": {
                "<covariate name>": ["<value 1>", "<value 2>", "..."]
            },
```

```
            "future_covariates": {
                "<covariate name>": ["<value 1>", "<value 2>", "..."]
            }
        }
    ],
    "options": {
        "rules": [
            {
                "covariate": "<future covariate name>",
                "operator": "==",
                "threshold": 1,
                "forecast_value": 0.0
            }
        ],
        "output_quantile": false,
        "point_forecast": "median",
        "quantile_levels": [0.1, 0.5, 0.9],
        "clip_min": 0,
        "clip_max": 100,
        "confidence_interval": 0.80,
        "cross_learning": false,
        "enhanced_forecast": false,
        "hierarchical_labels": [{"<key>": "<value>"}],
        "hierarchical_levels": [null, "<key>"]
    }
}'
```

Parameters

Top-Level and Per-Series Parameters

Name	Type	Description
`horizon`	integer	Required. Must be greater than 0. The number of future steps to forecast.
`data`	array	Required. A non-empty array of series objects. Each series is forecast independently unless `cross_learning` or `hierarchical_forecast` is used.
`data[].target`	array	Required. Per-series forecast target values. A list of lists for multivariate series, or a list of numbers for a single-variate series. `null` values are treated as `NaN`. Series longer than 8192 points are truncated to the most recent 8192.

Name	Type	Description
`data[].timestamp`	array	Optional. A list the same length as `target[i]`, containing at least 2 points. Supported formats: • `int` • `YYYY-MM-DD` • `YYYY-MM-DD HH:MM:SS` • `YYYY-MM-DD HH:MM:SS.<fraction>` (LONGDATE, fraction up to 7 digits) • `HH:MM:SS` (time-of-day, midnight rollover supported) Regularly-spaced timestamps are expected. A limited number of missing timestamps are supported. The service uses the smallest gap as the `detected_interval` and fills the unobserved positions before forecasting. Filled timestamps must not exceed 20% of the supplied timestamps, rounded up. `result_timestamp` is produced at the detected interval starting after the last observed timestamp. If omitted, `result_timestamp` and `detected_interval` are `null` for the series.
`data[].past_covariates`	object	Optional. An object mapping a covariate name to a list of values. Each list must match the length of `target[i]`. Values can be numeric, string, or mixed.

Name	Type	Description
`data[].future_covariates`	object	Optional. An object mapping a covariate name to a list of values. Each list must have a length equal to `horizon`. Each key must also exist in `past_covariates` of the same series.

`options` Parameters

The entire `options` block can be omitted. All fields use default values.

Name	Type	Default	Description
`rules`	array	`null`	Deterministic post-processing rules. For each position in the referenced future covariate array, if the value satisfies `<covariate> <operator> <threshold>`, the forecast at that position is overwritten by `forecast_value`. Supported operators are ==, !=, >=, <=, >, and <. If multiple rules match the same position, the last matching rule is applied. Rules take precedence over clipping.
`output_quantile`	Boolean	`false`	If set to `true`, returns all 21 quantiles in `quantile_result`. Otherwise, `quantile_result` is `null`. This value is always `null` for `hierarchical_forecast`.

Name	Type	Default	Description
`point_forecast`	string	`"median"`	Determines how `result` is calculated. • `"median"`: Uses Q0.5 of the quantile output. This option is MAE-optimal and is the default. • `"mean"`: Calculates the distribution mean using trapezoidal integration over the predicted quantiles. This option is MSE/RMSE-optimal and is recommended for intermittent demand. • `"hierarchical_forecast"`: Performs MinT-WLS reconciliation across aggregation levels. Requires `hierarchical_labels` and `hierarchical_levels`. Reconciled forecasts are returned in `hierarchical_result`, and `result` is returned as `[]`.
`quantile_levels`	array	`null`	Array of float values in the range (0,1). When specified, the selected levels are returned in `selected_quantile_result`. If `null`, the default 21-quantile grid is used. For `mean`, these levels also define the integration grid used to calculate the mean. Using more levels can improve accuracy.

Name	Type	Default	Description
`clip_min`	number	`null`	Clamps values in `result` that are below the specified value.
`clip_max`	number	`null`	Clamps values in `result` that are above the specified value. If both `clip_min` and `clip_max` are specified, `clip_min` must not be greater than `clip_max`.
`confidence_interval`	number	`0.80`	A value in the range (0,1) or `null`. If specified, `confidence_interval_output` returns the lower and upper quantile bounds for the interval. For example, a value of `0.80` returns q=0.10 and q=0.90 bounds. If set to `null`, `confidence_interval_output` is omitted. This value is always `null` for `hierarchical_forecast`.
`cross_learning`	Boolean	`false`	If set to `true`, all series in the request are treated as a single group and the model shares information across them. Results depend on the combination of series included in the request. No alignment, overlap, or equal-length requirements apply. If set to `false`, each series is forecast independently.

Name	Type	Default	Description
`enhanced_forecast`	Boolean	`false`	Applies additional inference techniques to improve forecast accuracy at the cost of increased inference time. Input time points are billed at a flat 4× rate when enabled. Output time points are never multiplied. For more information, see Best Practices.
`hierarchical_labels`	array	`null`	Array of `{string: string}` dictionaries. Defines per-series grouping labels for hierarchical reconciliation, with one dictionary for each input series (leaf). Required when `point_forecast` is set to `"hierarchical_forecast"`. Example: `[{"country":"germany","city":"berlin"}, ...]`

Name	Type	Default	Description
`hierarchical_levels`	array	`null`	Array of `null` or string values. Defines the aggregation levels to reconcile, ordered from the coarsest level to the finest level. Each entry must be either a label key defined in `hierarchical_labels` or `null` to represent the grand total. The returned `hierarchical_result` tree contains only the levels specified in this parameter. Label keys not specified do not appear as additional hierarchy levels. Required when `point_forecast` is set to `"hierarchical_forecast"`. Example: `[null, "country"]` This example produces a grand-total root node and country-level aggregates.

Response Payload

A successful request returns an HTTP 200 response with the following JSON structure. Array fields are shaped by the number of series, variates, and forecast horizon.

Response Example

```
{
    "result": [
        [
            ["<forecast value 1>", "<forecast value 2>", "..."]
        ]
    ],
    "quantile_result": null,
    "selected_quantile_result": null,
```

```
    "confidence_interval_output": [
        [
            [
                ["<lower bound 1>", "<lower bound 2>", "..."],
                ["<upper bound 1>", "<upper bound 2>", "..."]
            ]
        ]
    ],
    "result_timestamp": [
        ["<future timestamp 1>", "<future timestamp 2>", "..."]
    ],
    "detected_interval": [
        {
            "unit": "<interval unit>",
            "value": "<interval value>"
        }
    ],
    "all_null_targets": [],
    "hierarchical_result": null
}
```

The response can include the following parameters:

Name	Type	Description
`result`	3D array `[n_series][n_variates][horizon]`	Point forecast. For `median` and `mean`, the values are derived from the 21-quantile output. For `hierarchical_forecast`, this field always contains `[]`. Use `hierarchical_result` instead.
`quantile_result`	4D array `[n_series][n_variates][21][horizon]` or `null`	Returns all 21 fixed quantiles when `options.output_quantile` is `true`. Otherwise, the value is `null`. The value is always `null` for `hierarchical_forecast`. Supported quantile levels are: `[0.01, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95, 0.99]`

Name	Type	Description
`selected_quantile_result`	4D array `[n_series][n_variates][len(quantile_levels)][horizon]` or `null`	Returns quantile forecasts for the requested `quantile_levels`. If `quantile_levels` is `null`, the value is `null`. The value is always `null` for `hierarchical_forecast`.
`confidence_interval_output`	4D array `[n_series][n_variates][2][horizon]` or `null`	Contains lower and upper quantile bounds for the requested `confidence_interval`. If `confidence_interval` is `null`, the value is `null`. The value is always `null` for `hierarchical_forecast`. Index `0` contains the lower bound, and index `1` contains the upper bound. For example, `confidence_interval=0.80` returns q=0.10 as the lower bound and q=0.90 as the upper bound.
`result_timestamp`	Array of timestamp arrays or `null` per series	Future timestamps corresponding to forecast steps. The value is `null` for series that do not include `timestamp` values in the request. For `hierarchical_forecast`, the value is always `[]`.
`detected_interval`	Array of `{unit,value}` objects or `null` per series	Represents the interval inferred from the input timestamps. The value is `null` for series that do not include `timestamp` values in the request. For `hierarchical_forecast`, the value is always `[]`.

Name	Type	Description
`hierarchical_result`	Array of tree nodes or `null`	Contains hierarchical aggregate forecasts when `point_forecast` is set to `hierarchical_forecast`. For all other forecast modes, the value is `null`. Each node uses the format `{label_value:[[horizon_values]],"__children__":[...]}`. The forecast values use the shape `[n_variates][horizon]`. Deepest-level nodes omit the `__children__` field. If `hierarchical_levels` starts with `null`, the response contains a single root node whose key is an empty string. Otherwise, the response contains one node for each first-level aggregate, ordered alphabetically.
`all_null_targets`	Array of `{series_index,target_indices}`	Identifies non-empty target rows that contain only `null` values. Both indices are zero-based. The array is empty when no such rows exist.

Timestamp Handling

The `detected_interval.unit` value is determined by the input timestamp format.

Input Timestamp Format	`detected_interval.unit`	`result_timestamp` Format
`int`	`integer`	`int`
`YYYY-MM-DD` (non-monthly gaps)	`day`	`YYYY-MM-DD`
`YYYY-MM-DD` (calendar-monthly gaps)	`month`	`YYYY-MM-DD`
`YYYY-MM-DD HH:MM:SS` (non-monthly gaps)	`second`	`YYYY-MM-DD HH:MM:SS`

Input Timestamp Format	`detected_interval.unit`	`result_timestamp` Format
`YYYY-MM-DD HH:MM:SS` (calendar-monthly gaps)	`month`	`YYYY-MM-DD HH:MM:SS`
`YYYY-MM-DD HH:MM:SS.<fraction>` (LONG-DATE, non-monthly gaps)	`1e-7second`	`YYYY-MM-DD HH:MM:SS.<fraction>`
`YYYY-MM-DD HH:MM:SS.<fraction>` (LONG-DATE, calendar-monthly gaps)	`month`	`YYYY-MM-DD HH:MM:SS.<fraction>`
`HH:MM:SS` (time-of-day)	`second`	`HH:MM:SS`

> **Note**
>
> Timestamp strings are interpreted naively. The server does not perform timezone conversion.

Status Codes and Messages

The service returns standard HTTP status codes.

Status	Message	Description
`200`	OK	Successful forecast.
`400`	Bad Request	Business validation failed. The response can be caused by one of the following conditions: • Timestamp and target length mismatch. • Unsupported timestamp format. • Too many missing timestamps to fill (more than 20% of the supplied timestamps). • A `future_covariates` key that doesn't exist in `past_covariates`. Response body: `{"detail":"<error message>"}`

Status	Message	Description
422	Unprocessable Content	Schema validation failed. The response can be caused by one of the following conditions: • Missing required field. • Wrong data type. • Value out of range. • Unknown field. Response body (FastAPI standard): `{ "detail": [ { "loc": [...], "msg": "...", "type": "..." } ] }`
500	Internal Server Error	An unexpected error occurred. The response can be caused by one of the following conditions: • The model crashed. • The pipeline wasn't initialized. Response body: `{ "detail": "Internal server error" }`

1.3.9.1.4.2 Sample Inference Requests and Responses

The following examples demonstrate common forecasting scenarios and show how different request parameters affect the response returned by the inference service.

Successful Prediction (200): Request with Integer Timestamps

This example uses integer timestamps.

Request

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'content-type: application/json' \
  --data '{"horizon":2,"data":[{"timestamp":[1,2,3,4,5,6,7,8,9,10],"target":
[[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0]]}]}'
```

Response

Output Code


```
{"result":
[[[10.996417999267578,11.85465145111084]]],"quantile_result":null,"selected_qu
antile_result":null,"confidence_interval_output":
[[[[10.568910598754883,11.262651443481445],
[11.472352027893066,12.359424591064453]]]],"result_timestamp":
[[11,12]],"detected_interval":
[{"unit":"integer","value":1}],"all_null_targets":
[],"hierarchical_result":null,"group_results":null}
```


Successful Prediction (200): Request with Date Timestamps

This example uses date-based timestamps.

Request

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'content-type: application/json' \
  --data '{"horizon":3,"data":[{"timestamp":
["2020-01-01","2020-01-02","2020-01-03","2020-01-04","2020-01-05","2020-01-06","2
020-01-07","2020-01-08","2020-01-09","2020-01-10"],"target":
[[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0]]}]}'
```

Response

Output Code


```
{"result":
[[[10.996417999267578,11.85465145111084,12.698359489440918]]],"quantile_result
":null,"selected_quantile_result":null,"confidence_interval_output":
[[[[10.568910598754883,11.262651443481445,12.033546447753906],
[11.472352027893066,12.359424591064453,13.541707992553711]]]],"result_timestam
p":[["2020-01-11","2020-01-12","2020-01-13"]],"detected_interval":
[{"unit":"day","value":1}],"all_null_targets":
[],"hierarchical_result":null,"group_results":null}
```

Successful Prediction (200): Request with SECONDDATE Timestamps

This example uses SECONDDATE timestamps.

Request

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'content-type: application/json' \
  --data '{"horizon":2,"data":[{"timestamp":["2020-01-01
00:00:01.001","2020-01-01 00:00:01.002","2020-01-01 00:00:01.003","2020-01-01
00:00:01.004","2020-01-01 00:00:01.005","2020-01-01 00:00:01.006","2020-01-01
00:00:01.007","2020-01-01 00:00:01.008","2020-01-01 00:00:01.009","2020-01-01
00:00:01.010"],"target":[[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0]]}]}'
```

Response

Output Code

```
{"result":
[[[10.996417999267578,11.85465145111084]]],"quantile_result":null,"selected_qu
antile_result":null,"confidence_interval_output":
[[[[10.568910598754883,11.262651443481445],
[11.472352027893066,12.359424591064453]]]],"result_timestamp":[["2020-01-01
00:00:01.011","2020-01-01 00:00:01.012"]],"detected_interval":
[{"unit":"1e-7second","value":10000}],"all_null_targets":
[],"hierarchical_result":null,"group_results":null}
```

Successful Prediction (200): Request with LONGDATE Timestamps

This example uses LONGDATE timestamps.

Request

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'content-type: application/json' \
  --data '{"horizon":2,"data":[{"timestamp":["2020-01-01
00:00:01.001","2020-01-01 00:00:01.002","2020-01-01 00:00:01.003","2020-01-01
00:00:01.004","2020-01-01 00:00:01.005","2020-01-01 00:00:01.006","2020-01-01
00:00:01.007","2020-01-01 00:00:01.008","2020-01-01 00:00:01.009","2020-01-01
00:00:01.010"],"target":[[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0]]}]}'
```

Response

Output Code

```
{"result":
[[[10.996417999267578,11.85465145111084]]],"quantile_result":null,"selected_qu
antile_result":null,"confidence_interval_output":
```

```
[[[[10.568910598754883,11.262651443481445],
[11.472352027893066,12.359424591064453]]]],"result_timestamp":[["2020-01-01
00:00:01.011","2020-01-01 00:00:01.012"]],"detected_interval":
[{"unit":"1e-7second","value":10000}],"all_null_targets":
[],"hierarchical_result":null,"group_results":null}
```


Successful Prediction (200): Request with output_quantile

Returns all 21 quantiles in `quantile_result`.

Request

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'content-type: application/json' \
  --data '{"horizon":2,"data":[{"timestamp":[1,2,3,4,5,6,7,8,9,10],"target":
[[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0]]}],"options":
{"output_quantile":true}}'
```

Response

Output Code


```
{"result":[[[10.996417999267578,11.85465145111084]]],"quantile_result":
[[[[7.5556230545043945,7.744816780090332],
[10.193599700927734,10.68617057800293],
[10.568910598754883,11.262651443481445],[10.570749282836914,11.3654203414917],
[10.743620872497559,11.486021041870117],
[10.795597076416016,11.586033821105957],
[10.871317863464355,11.722585678100586],
[10.889595031738281,11.743776321411133],
[10.926765441894531,11.785883903503418],
[10.986242294311523,11.841873168945312],
[10.996417999267578,11.85465145111084],
[11.025259017944336,11.946949005126953],
[11.073755264282227,11.965944290161133],
[11.138389587402344,12.05129623413086],
[11.204987525939941,12.129083633422852],
[11.208433151245117,12.215331077575684],
[11.261232376098633,12.271505355834961],
[11.352106094360352,12.344038009643555],
[11.472352027893066,12.359424591064453],
[11.682449340820312,12.531235694885254],
[12.57781982421875,13.738334655761719]]]],"selected_quantile_result":null,"con
fidence_interval_output":[[[[10.568910598754883,11.262651443481445],
[11.472352027893066,12.359424591064453]]]],"result_timestamp":
[[11,12]],"detected_interval":
[{"unit":"integer","value":1}],"all_null_targets":
[],"hierarchical_result":null,"group_results":null}
```

Successful Prediction (200): Request with a Custom confidence_interval

Sets a 95% interval. The bounds in `confidence_interval_output` are the q=0.025 and q=0.975 quantiles (index 0 is the lower bound, index 1 the upper), which are wider than the default 80% interval.

Request

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'content-type: application/json' \
  --data '{"horizon":3,"data":[{"timestamp":[1,2,3,4,5,6,7,8,9,10],"target":
[[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0]]}],"options":
{"confidence_interval":0.95}}'
```

Response

Output Code


```
{"result":
[[[10.996417999267578,11.85465145111084,12.698359489440918]]],"quantile_result
":null,"selected_quantile_result":null,"confidence_interval_output":
[[[[8.544864654541016,8.847824096679688,9.358936309814453],
[12.242055892944336,13.285673141479492,14.644306182861328]]]],"result_timestam
p":[[11,12,13]],"detected_interval":
[{"unit":"integer","value":1}],"all_null_targets":
[],"hierarchical_result":null,"group_results":null}
```


Successful Prediction (200): Request with point_forecast: mean

This example uses `point_forecast` set to `mean`.

Request

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'content-type: application/json' \
  --data '{"horizon":5,"data":[{"timestamp":[1,2,3,4,5,6,7,8,9,10],"target":
[[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0]]}],"options":
{"point_forecast":"mean"}}'
```

Response

Output Code


```
{"result":
[[[11.013999938964844,11.864879608154297,12.777382850646973,13.760469436645508
,14.703609466552734]]],"quantile_result":null,"selected_quantile_result":null,
"confidence_interval_output":
[[[[10.568910598754883,11.262651443481445,12.033546447753906,12.75979232788086
```

```
,13.503268241882324],
[11.472352027893066,12.359424591064453,13.541707992553711,14.721451759338379,1
5.873589515686035]]]],"result_timestamp":
[[11,12,13,14,15]],"detected_interval":
[{"unit":"integer","value":1}],"all_null_targets":
[],"hierarchical_result":null,"group_results":null}
```


Successful Prediction (200): Request with point_forecast: mean and Custom quantile_levels

Uses trapezoidal integration over 9 quantile levels to compute the distribution mean, recommended for intermittent-demand forecasting. When `quantile_levels` is set, the same levels are returned in `selected_quantile_result`.

Request

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'content-type: application/json' \
  --data '{"horizon":10,"data":[{"timestamp":
["2007-12-10","2007-12-12","2007-12-14","2007-12-16","2007-12-18","2008-01-15","2
008-01-17","2008-01-19","2008-01-21","2008-01-23","2008-01-25","2008-01-27","2008
-01-29","2008-02-02","2008-02-04","2008-02-06","2008-02-08","2008-02-10","2008-02
-12","2008-02-14"],"target":
[[10.25,11.99,13.27,13.13,11.02,10.43,12.23,13.88,15.33,16.72,15.64,14.4,14.17,13
.98,16.14,18.78,19.44,19.52,17.68,16.84]],"past_covariates":{"airtemperature":
[15.44,17.28,17.95,15.87,13.54,12.11,12.31,15.11,17.17,17.84,16.12,14.25,12.07,12
.34,15.25,16.98,17.99,15.71,13.3,12.13]},"future_covariates":{"airtemperature":
[12.88,15.05,17.31,17.83,15.86,13.48,11.94,12.97,15.1,16.82]}}],"options":
{"point_forecast":"mean","quantile_levels":
[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]}}'
```

Response

Output Code


```
{"result":
[[[17.364219665527344,18.583675384521484,20.325830459594727,20.580018997192383
,19.99493408203125,18.86580467224121,18.245637893676758,18.469654083251953,19.
736997604370117,21.58102035522461]]],"quantile_result":null,"selected_quantile
_result":
[[[[15.861515998840332,16.566572189331055,17.601001739501953,17.5913143157959,
16.712663650512695,15.692886352539062,14.958240509033203,14.887096405029297,15
.703776359558105,16.815309524536133],
[16.54750633239746,17.45974349975586,18.831235885620117,18.918731689453125,18.
04950714111328,16.962177276611328,16.238269805908203,16.250152587890625,17.206
666946411133,18.629743576049805],
[16.925079345703125,17.97675323486328,19.4957275390625,19.65349578857422,18.86
7145538330078,17.7641658782959,17.089292526245117,17.15414047241211,18.2417335
5102539,19.864036560058594],
[17.196815490722656,18.338701248168945,20.019752502441406,20.14012908935547,19
.521574020385742,18.366859436035156,17.71973419189453,17.89889907836914,19.033
889770507812,20.79290771484375],
[17.415781021118164,18.653329849243164,20.379512786865234,20.599533081054688,2
0.037200927734375,18.89378547668457,18.311809539794922,18.51116943359375,19.72
```

```
9907989501953,21.644052505493164],
[17.636810302734375,18.955204010009766,20.824752807617188,21.066661834716797,2
0.60116195678711,19.432884216308594,18.835208892822266,19.10914421081543,20.47
6547241210938,22.44400978088379],
[17.886484146118164,19.31573486328125,21.328678131103516,21.669452667236328,21
.163698196411133,20.040342330932617,19.47285270690918,19.830299377441406,21.19
437026977539,23.298524856567383],
[18.168689727783203,19.733245849609375,21.82098388671875,22.306446075439453,21
.927082061767578,20.73362922668457,20.21141815185547,20.625232696533203,22.266
944885253906,24.463993072509766],
[18.715164184570312,20.369441986083984,22.770782470703125,23.372516632080078,2
3.141986846923828,21.949920654296875,21.426950454711914,21.991235733032227,23.
776165008544922,26.299983978271484]]]],"confidence_interval_output":
[[[[15.861515998840332,16.566572189331055,17.601001739501953,17.5913143157959,
16.712663650512695,15.692886352539062,14.958240509033203,14.887096405029297,15
.703776359558105,16.815309524536133],
[18.715164184570312,20.369441986083984,22.770782470703125,23.372516632080078,2
3.141986846923828,21.949920654296875,21.426950454711914,21.991235733032227,23.
776165008544922,26.299983978271484]]]],"result_timestamp":
[["2008-02-16","2008-02-18","2008-02-20","2008-02-22","2008-02-24","2008-02-26
","2008-02-28","2008-03-01","2008-03-03","2008-03-05"]],"detected_interval":
[{"unit":"day","value":2}],"all_null_targets":
[],"hierarchical_result":null,"group_results":null}
```


Successful Prediction (200): Request with enhanced_forecast

This example enables `enhanced_forecast` to generate improved forecast predictions.

Request

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'content-type: application/json' \
  --data '{"horizon":2,"data":[{"timestamp":[1,2,3,4,5,6,7,8,9,10],"target":
[[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0]]}],"options":
{"enhanced_forecast":true}}'
```

Response

Output Code


```
{"result":
[[[10.990999221801758,11.929146766662598]]],"quantile_result":null,"selected_q
uantile_result":null,"confidence_interval_output":
[[[[10.529675483703613,11.302016258239746],
[11.476777076721191,12.441227912902832]]]],"result_timestamp":
[[11,12]],"detected_interval":
[{"unit":"integer","value":1}],"all_null_targets":
[],"hierarchical_result":null,"group_results":null}
```

Successful Prediction (200): Request with cross_learning

Use when series are from related domains and cross-series information would be signal rather than noise.

Request

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'content-type: application/json' \
  --data '{"horizon":2,"data":[{"timestamp":[1,2,3,4,5,6,7,8,9,10],"target":
[[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0]]},{"timestamp":
[1,2,3,4,5,6,7,8,9,10],"target":
[[10.0,9.0,8.0,7.0,6.0,5.0,4.0,3.0,2.0,1.0]]}],"options":
{"cross_learning":true}}'
```

Response

Output Code


```
{"result":[[[10.75124454498291,11.541213989257812]],
[[0.1672377586364746,0.0]]],"quantile_result":null,"selected_quantile_result":
null,"confidence_interval_output":[[[[10.398977279663086,11.130542755126953],
[11.103979110717773,12.020644187927246]]],[[[0.0,0.0],
[0.6594223976135254,0.0]]]],"result_timestamp":[[11,12],
[11,12]],"detected_interval":[{"unit":"integer","value":1},
{"unit":"integer","value":1}],"all_null_targets":
[],"hierarchical_result":null,"group_results":null}
```


Successful Prediction (200): Request with point_forecast: hierarchical_forecast

Six leaf series carry `country`, `city`, and `district` labels, but `hierarchical_levels` requests reconciliation only at `["country", "city"]`. The response is a tree containing exactly those two levels: country nodes with city children. The unlisted `district` key does not add further depth. No timestamps are provided, so `result_timestamp` and `detected_interval` are empty.

Request

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'content-type: application/json' \
  --data '{"horizon":4,"data":[{"target":[[10,11,12,13,14,15]]},{"target":
[[8,9,10,11,12,13]]},{"target":[[6,7,8,9,10,11]]},{"target":[[5,6,7,8,9,10]]},
{"target":[[4,5,6,7,8,9]]},{"target":[[3,4,5,6,7,8]]}],"options":
{"point_forecast":"hierarchical_forecast","hierarchical_labels":
[{"country":"germany","city":"berlin","district":"mitte"},
{"country":"germany","city":"berlin","district":"charlottenburg"},
{"country":"germany","city":"hamburg","district":"altona"},
{"country":"germany","city":"hamburg","district":"harburg"},
{"country":"france","city":"paris","district":"montmartre"},
```

```
{"country":"france","city":"paris","district":"marais"}],"hierarchical_levels":
["country","city"]}}'
```

Response

Output Code

```
{"result":
[],"quantile_result":null,"selected_quantile_result":null,"confidence_interval
_output":null,"result_timestamp":[],"detected_interval":[],"all_null_targets":
[],"hierarchical_result":[{"france":
[[18.548839569091797,20.04424476623535,21.7530460357666,23.45072364807129]],"_
_children__":[{"paris":
[[52.097679138183594,55.08848476409912,58.506086349487305,61.90144443511963]]}
]},{"germany":
[[29.548839569091797,31.044241905212402,32.753045082092285,34.45072364807129]]
,"__children__":[{"berlin":
[[22.548839569091797,24.04424285888672,25.75304126739502,27.45072078704834]]},
{"hamburg":
[[18.548839569091797,20.04424476623535,21.7530460357666,23.45072364807129]]}]}
],"group_results":null}
```

Successful Prediction (200): Request with Clipping

This example applies clipping to constrain forecast values between the specified `clip_min` and `clip_max` bounds.

Request

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'content-type: application/json' \
  --data '{"horizon":3,"data":[{"timestamp":[1,2,3,4,5,6,7,8,9,10],"target":
[[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0]]}],"options":
{"clip_min":0,"clip_max":11}}'
```

Response

Output Code

```
{"result":
[[[10.996417999267578,11.0,11.0]]],"quantile_result":null,"selected_quantile_r
esult":null,"confidence_interval_output":
[[[[10.568910598754883,11.262651443481445,12.033546447753906],
[11.472352027893066,12.359424591064453,13.541707992553711]]]],"result_timestam
p":[[11,12,13]],"detected_interval":
[{"unit":"integer","value":1}],"all_null_targets":
[],"hierarchical_result":null,"group_results":null}
```

Successful Prediction (200): Request with Deterministic Rules

Overwrites the forecast at every future step where the `is_holiday` covariate equals `1`.

Request

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'content-type: application/json' \
  --data '{"horizon":3,"data":[{"timestamp":[1,2,3,4,5,6,7,8,9,10],"target":
[[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0]],"past_covariates":{"is_holiday":
[0,0,0,0,0,0,0,0,0,0]},"future_covariates":{"is_holiday":[0,1,0]}}],"options":
{"rules":
[{"covariate":"is_holiday","operator":"==","threshold":1,"forecast_value":0.0}]}}
'
```

Response

Output Code


```
{"result":
[[[11.048547744750977,0.0,12.874539375305176]]],"quantile_result":null,"select
ed_quantile_result":null,"confidence_interval_output":
[[[[10.74734878540039,11.256742477416992,12.25273609161377],
[11.459436416625977,12.39759635925293,13.73572063446045]]]],"result_timestamp"
:[[11,12,13]],"detected_interval":
[{"unit":"integer","value":1}],"all_null_targets":
[],"hierarchical_result":null,"group_results":null}
```


Unprocessable Content (422): Missing Required horizon

This example omits the required `horizon` field from the request body.

Request

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'content-type: application/json' \
  --data '{"data":[{"timestamp":["2020-01-01","2020-01-02"],"target":
[[1.0,2.0]]}]}'
```

Response

Output Code


```
{"detail":[{"type":"missing","loc":["body","horizon"],"msg":"Field
required","input":{"data":[{"timestamp":["2020-01-01","2020-01-02"],"target":
[[1.0,2.0]]}]}}]}
```

Bad Request (400): Invalid Timestamp Format

This example uses timestamps in the unsupported `MM/DD/YYYY` format. The `timestamp` field must use a supported format such as `YYYY-MM-DD`, `YYYY-MM-DD HH:MM:SS`, `YYYY-MM-DD HH:MM:SS[.fraction]`, `HH:MM:SS`, or an integer value.

Request

```
curl --request POST \
  --url "$DEPLOYMENT_URL/predict" \
  --header 'AI-Resource-Group: default' \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header 'content-type: application/json' \
  --data '{
    "horizon": 2,
    "data": [
      {
        "timestamp": ["01/01/2020", "01/02/2020", "01/03/2020"],
        "target": [[1.0, 2.0, 3.0]]
      }
    ]
  }'
```

Response

Output Code

```
{
  "detail": "timestamp must match one supported format: YYYY-MM-DD, YYYY-MM-
DD HH:MM:SS, YYYY-MM-DD HH:MM:SS[.fraction], HH:MM:SS, or INTEGER"
}
```

Note

The request fails because the timestamps use the unsupported `MM/DD/YYYY` format. Use a supported timestamp format, for example, `2020-01-01`.

1.3.9.1.5 SAP-RPT-Decisions

For general information about the SAP-RPT model family, including key features, tabular in-context learning, deployment instructions, API usage, and response formats, see SAP-RPT-1 [page 121] overview.

Model Specifications

Value	sap-rpt-decisions
Max Examples	2048
Max Questions	10
Max Named State Fields	99
Options / Levels per Question	255
Recommended Number of Examples	500-2000 [1]

[1] Generic recommendations on the number of examples are derived from SAP's own experimentation and represent best-practice values that work well for many considered use cases, allowing for a balanced trade-off between prediction quality, model runtime and costs. Individual application scenarios may however achieve best outcomes at different settings or may work with fewer examples, also depending on the customer's individual requirements. For best results, SAP recommends own experimentation by the customer or implementation partner in the given application scenario setting with productive data.

Decision API (`/systemone`)

The `/systemone` endpoint answers *decision* tasks. You can describe what to evaluate (`state`) and ask **one or more questions** about it, and the service returns a typed answer per question.

Because the model learns in context, it acts as a **few-shot learner**, not a zero-shot learner. You must pass a list of at least two few-shot `examples`: labeled cases, each shaped like `state` plus the known answers. The examples are what the model learns from, and `state` is the single case it answers for. More and better-matched examples give better answers. Examples may be sparse, asa field present in some examples and absent in others is treated as a missing value. All possible answers (for example, choices or scores) must be demonstrated in the input `examples`.

A request is built from three parts:

- `state`: the single case to evaluate.
- `questions`: one or more typed questions to answer about that `state`.
- `examples`: labeled cases the model learns from.

Limits

The following limits apply to a single Decision API request (see also the table above):

- **Examples**: at least 2 are required and up to 2048 are accepted, with 500–2000 recommended for the best balance of quality, latency and cost.
- **Questions**: up to 10 per request, each answered for the same `state`.
- **Options or levels**: each `choice` option set and each `score` scale can have up to 255. Each option/level must appear in at least one provided example.

- **State fields**: the number of named `state` fields plus the number of questions must stay within 100. A plain-string or array `state` counts as a single field, so in practice only large structured states approach this limit.

A single request evaluates exactly one `state`. To score many cases, send one request per `state`, reusing the same `examples`.

Request Payload

name	type	description
`state`	string / object / array	The thing to evaluate. A plain string, structured object (named fields), or array of text values (for example, a chat log).
`model`	string	Optional. The model name, included for consistency with other SAP-RPT endpoints. It has no effect on the result; the response reports the model that served the request.
`questions`	object	Map of question key to a typed question (`noul`, `choice`, or `score`). At least one is required.
`questions.*.type`	string	`"noul"`, `"choice"`, or `"score"`. For more information about the per type `criteria` shape, see the following *Question Types* information.
`questions.*.instructions`	string / object / array	Describes the decision to make. This frames the question, so phrase it precisely.
`questions.*.criteria`	object / array	Describes the possible answers. Its shape depends on `type`. For more information about the per type `criteria` shape, see the following *Question Types* information. Filling in the descriptions gives a more accurate answer.

name	type	description
`examples`	array	**Required; at least two.** Labeled few-shot cases. Each is `{"state": ..., "answers": {"<question key>": <known answer>}}`, with each example `state` shaped like the request `state` (same fields). An example need not answer every question. Every option/level/Boolean value you declare must appear as the answer in at least one example, asthe model can only predict labels it has seen in context.

Question types

Each question has a `type` that fixes the shape of its `criteria` (in the request) and of its answer (in the response). The three types:

type	what it asks	answer fields
`noul`	a yes/no (Boolean) judgement	`noul`: probability the statement is true (0 to 1)
`choice`	pick one of several named options	`choice` The following values are supported: • `probabilities` • `confidence`
`score`	rate on an ordered scale (lowest first)	`score` (probability-weighted mean level), `legend`, `probabilities`, `confidence`

The subsections below cover the request side of each type (its `criteria` and how to encode example answers). For more information about the answer fields, see the **Response Payload** section below..

`noul`: Boolean Judgement

A yes/no question. `criteria` is optional; when provided, it describes what each outcome means to sharpen the answer:

```
"is_negative": {
    "type": "noul",
    "instructions": "Is this review negative?",
    "criteria": { "true": "Expresses dissatisfaction", "false": "Neutral or
positive" }
}
```

In `examples`, answer with a Boolean: `true` or `false`. Include at least one example of each outcome you expect the model to predict.

`choice`: Pick one Option

Pick exactly one of several named options. `criteria` is a {option: description} map with at least two options:

```
"topic": {
    "type": "choice",
    "instructions": "Which topic does this review concern?",
    "criteria": {
        "Battery": "Charge, runtime or power issues",
        "Support": "Customer service and responsiveness",
        "Price": "Cost or value for money"
    }
}
```

In `examples`, answer with the selected option as a string, matching one of the `criteria` keys exactly. Every option must appear as the answer in at least one example.

`score`: Rate on an Ordered Scale

Rate on an ordered scale. `criteria` is a list of level descriptions, **lowest first**, with at least two levels:

```
"satisfaction": {
    "type": "score",
    "instructions": "Rate the customer satisfaction.",
    "criteria": ["Very unhappy", "Unhappy", "Neutral", "Happy", "Very happy"]
}
```

In `examples`, answer with the **0-based index** of the level (for example, `0` for the first/lowest level, `4` for the fifth), not the level's text. Every level index must appear as the answer in at least one example.

Response payload

The response holds an `answers` object keyed by the same question keys you sent. Each answer carries its question's `type` plus the fields for that type:

field	present for	description
`noul`	`noul`	Probability the statement is true (0 to 1).
`choice`	`choice`	The selected option (one of the `criteria` keys).
`score`	`score`	Probability-weighted mean of the level indices. A value of `0.54` lies between levels `0` and `1`.
`legend`	`score`	Maps each level index (as a string) to its description, so you can turn `score` back into labels.
`probabilities`	`choice`, `score`	Distribution over the options (`choice`) or levels (`score`); sums to 1.
`confidence`	`choice`, `score`	How decisive the distribution is, on a 0 to 1 scale.

`noul`

A `noul` answer carries only the `noul` probability, asthere is no separate `confidence` or `probabilities` field, since the single value already captures P(true).

`confidence`

`confidence` is derived from the full `probabilities` distribution, asit is not simply the largest probability. It is scaled so that a flat distribution, which carries no information, reads `0`, and a distribution that puts all its mass on a single answer reads `1`. For a `score` question it also accounts for the ordering of the levels: probability mass spread across neighbouring levels lowers confidence less than the same mass spread across distant ones, since adjacent levels still agree on roughly where the answer sits. So two answers can share the same top probability yet report different `confidence`.

Example

A request may ask several questions of different types at once. This one asks a `noul`, a `choice`, and a `score` question about the same `state`:

```
curl --request POST \
  --url "$DEPLOYMENT_URL/systemone" \
  --header "AI-Resource-Group: default" \
  --header "Authorization: Bearer $AUTH_TOKEN" \
  --header "Content-Type: application/json" \
  --data '{
    "model": "sap-rpt-decisions",
    "state": { "review": "Battery dies in an hour and support never replied." },
    "questions": {
        "is_negative": {
            "type": "noul",
            "instructions": "Is this review negative?"
        },
        "topic": {
            "type": "choice",
            "instructions": "Which topic does this review concern?",
            "criteria": {
                "Battery": "Charge, runtime or power issues",
                "Support": "Customer service and responsiveness",
                "Price": "Cost or value for money"
            }
        },
        "satisfaction": {
            "type": "score",
            "instructions": "Rate the customer satisfaction.",
            "criteria": ["Very unhappy", "Unhappy", "Neutral", "Happy", "Very
happy"]
        }
    },
    "examples": [
        {
            "state": { "review": "Love it, works perfectly and the team was
super helpful." },
            "answers": { "is_negative": false, "topic": "Support",
"satisfaction": 4 }
        },
        {
            "state": { "review": "Too expensive for what it does." },
            "answers": { "is_negative": true, "topic": "Price", "satisfaction":
1 }
        },
```

```
        {
            "state": { "review": "Battery drains fast but otherwise fine." },
            "answers": { "is_negative": true, "topic": "Battery",
"satisfaction": 2 }
        },
        {
            "state": { "review": "Completely useless, broke on day one and
nobody helped." },
            "answers": { "is_negative": true, "topic": "Support",
"satisfaction": 0 }
        },
        {
            "state": { "review": "Decent value and the battery lasts a full
day." },
            "answers": { "is_negative": false, "topic": "Price", "satisfaction":
3 }
        }
    ]
}'
```

Response (one answer per question, keyed by the question key):

```
{
  "model": "sap-rpt-decisions",
  "answers": {
    "is_negative": { "type": "noul", "noul": 0.91 },
    "topic": {
      "type": "choice",
      "choice": "Battery",
      "probabilities": { "Battery": 0.74, "Support": 0.21, "Price": 0.05 },
      "confidence": 0.61
    },
    "satisfaction": {
      "type": "score",
      "score": 0.54,
      "legend": { "0": "Very unhappy", "1": "Unhappy", "2": "Neutral", "3":
"Happy", "4": "Very happy" },
      "probabilities": { "0": 0.6, "1": 0.3, "2": 0.07, "3": 0.02, "4": 0.01 },
      "confidence": 0.46
    }
  },
  "usage": { "input_tokens": 21, "output_tokens": 3 },
  "metadata": { "num_columns": 4, "num_rows": 6, "num_predictions": 3,
"num_query_rows": 1 }
}
```

Handling Long-Tailed Regression Target Values

A log transform is a useful default for positive, long-tailed targets when a similar percentage error should have a similar impact across small and large values. By compressing the target range, a log transform commonly improves relative accuracy for small and medium values.

For strictly positive targets, use `log(y)` and transform predictions back to the original scale with `exp`.

For non-negative targets that can contain zero values, use `log1p(y)` and transform predictions back with `expm1`.

> **Note**
>
> A log transformation isn't universally better for all use cases. Choose the target scale based on the business cost of prediction errors.

For example, a 1% error on a value of 1,000,000 can be more costly than a 100% error on a value of 1,000. If large absolute errors or rare, high-value outcomes are costly, retain the original target scale.

The distinction is based on the cost of an error rather than the semantic type of the data. For example, currency values can reasonably use either the original scale or a log-transformed scale depending on the business objective.

Comparing Target Scales

If the appropriate error cost is unclear or the decision is important, evaluate both the original and log-transformed target scales on representative holdout data.

When comparing results, transform predictions generated on the log scale back to the original unit before evaluating business metrics.

Analyze performance separately for low, medium, and high target ranges to understand how each target scale affects prediction quality across the distribution.

> ⓘ Note
>
> Avoid using unqualified MAPE when actual target values can be zero or close to zero, as the metric can become unstable or misleading.

When available, review prediction intervals or quantile estimates in addition to point predictions, especially when rare, high-value outcomes have significant business impact.

1.3.9.2 SAP-ABAP

1.3.9.2.1 SAP-ABAP-1

SAP-ABAP-1 is a foundation model built by SAP that helps with coding tasks. It is fine-tuned on a large amount of SAP ABAP code.

Key Features

Key features include:

- **Pretrained Model:** The model is fine-tuned on the SAP S/4HANA ABAP and CDS code base. There's no need to collect, prepare, or manage training data. You also don't need to manage costly compute infrastructure for training.
- **Code Explanation:** The model helps you to understand code by turning code to text. For more information, see Using SAP-ABAP-1 to Explain Code [page 176].
- **Integration:** You can integrate the model into development tools, IDEs, or agentic frameworks.
- **Access to SAP Internal Models:** You gain direct access to the model used internally for the SAP Joule for developers feature in ADT. The same model is also used for SAP Joule for Consultants and the Custom

Code Migration scenario, which explains old code to users during the migration from SAP ERP to SAP S/4HANA.

Accessing SAP-ABAP-1

`SAP-ABAP-1` can only be accessed via the Orchestration service of the generative AI hub. Make sure you have met the following prerequisites before you start:

- You are onboarded to SAP AI Core and have an SAP AI Core service instance and service key. For more information, see Initial Setup.
- You are using the extended service plan. For more information, see Service Plans and Update a Service Plan.
- You have completed the client authorization for your preferred user interface. For more information, see Use a Service Key.
- You have obtained your Orchestration service deployment URL. For more information, see Get An Orchestration Deployment URL [page 7].

You can submit a test call to `SAP-ABAP-1` using the Orchestration service. To do so, use `sap-abap-1` as the `model_name` in the request payload.

Using SAP-ABAP-1 to Explain Code

After you have met the prerequisites, you can make inference calls. `SAP-ABAP-1` supports the code-to-text scenario. The input is an ABAP class or a code snippet from your ABAP code repository. The output is an explanation in English of this class or code snippet.

For more information about how to make inference calls, see Example Payloads for Inferencing: SAP-ABAP-1 [page 177].

Context Length

The maximum context length is 128,000 tokens. Considering the need to reserve some context length for prompt instructions and output, the maximum input code size is approximately 120,000 tokens. This corresponds to roughly 250,000 input code characters.

Input/Output Filtering

Input and output filtering is configured for the model to prevent malicious or harmful content being included in the input or output.

FAQ

1. **What is the relationship between `SAP-ABAP-1` and the SAP Joule for developers feature in the ABAP Development Tools for Eclipse?**
   `SAP-ABAP-1` is used internally for the SAP Joule for developers feature in the ABAP Development Tools for Eclipse.

2. **Can** SAP-ABAP-1 **also be used to generate code?**
   The current version of SAP-ABAP-1 is optimized for explaining ABAP code. It can also generate code, but this capability is experimental and not recommended for productive use. You're welcome to explore code generation, but please note that it's not the primary focus of the model.
3. **Can the model be integrated into the ABAP Development Tools for Eclipse, VS Code, or other IDEs?**
   The model is natively integrated into the ABAP Development Tools for Eclipse (ADT) as part of SAP Joule for developers. Currently, there isn't an out-of-the-box integration for VS Code. However, it's technically possible to integrate with any IDE that has extension interfaces.
4. **Is it possible to access the model weights for further fine-tuning?**
   No, currently SAP-ABAP-1 is only accessible through the Orchestration service of the generative AI hub.
5. **Can I test** SAP-ABAP-1 **for free?**
   Yes, you can test the model for 30 days free of charge by using the "Try now" feature. For more information, see Generative AI Hub.

Related Information

Example Payloads for Inferencing: SAP-ABAP-1 [page 177]

1.3.9.2.1.1 Example Payloads for Inferencing: SAP-ABAP-1

Inferencing on the SAP-ABAP-1 model through the generative AI hub Orchestration service can be done with a POST HTTP call to the `ORCHESTRATION_SERVICE` endpoint with an appropriate payload and API headers. For information about the headers, request payload, and expected response, see Orchestration Workflow V2 [page 382].

The Orchestration service offers multiple configuration options. From these options, `prompt_templating` and model configuration are mandatory.

> → Tip
>
> If you use a Windows device, use Windows PowerShell, and replace `curl` with `curl.exe`.

Best Practices

- Code with syntax errors or missing dependencies produces poor results. Use the model on finished code instead.
- Including inline code documentation enhances outcomes.
- If you're interested in understanding a specific part of your class, provide only that section as input. Supplying very long classes in their entirety may lead to sub-optimal results.
- Avoid inputting commented-out code because it can confuse the model.
- Only input ABAP code – other programming languages aren't supported.

Request Payloads

The curl script below shows a typical sample call to explain an ABAP class.

```
curl --request POST \
  --url "$ORCHESTRATION_SERVICE_URL/v2/completion" \
  --header 'ai-resource-group: default' \
  --header "authorization: Bearer $AUTH_TOKEN" \
  --header 'Content-Type: application/json' \
  --data '{
  "config": {
    "modules": {
      "prompt_templating": {
        "prompt": {
          "template": [
            {
              "role": "user",
              "content": "Explain the ABAP class in the input code below.\nThe
explanation should have the following five sections. Do not provide a header
for the sections.\n- Introduce and summarize the class at the beginning in
one paragraph.\n- Provide a summary of the variables, without going into too
much detail and provide individual headers. The overall header should be
named `Data Declaration´.\n- If no interfaces are implemented in the class,
skip this section. Provide a summary of the interfaces, without going into
too much detail and provide individual headers. The overall header should be
named `Interfaces´.\n- Provide a summary of each method, without going into
too much detail and provide individual headers. The overall header should
be named `Methods´.\n- Give a brief conclusion that focuses on the usage
of the class. Do not provide an overall header for this section.\nConsider
these guidelines:Focus on dependencies in the code and the side effects
of the codes.Use variable names when needed.\nMention method calls when
needed.\nExplain local includes when needed.\nReject hate speech.\nUse the
following extra context section if provided as additional information to explain
the given code.\n## Input code:\n```abap\nCLASS zcl_add_test DEFINITION\n
PUBLIC\n  FINAL\n  CREATE PUBLIC .\n\n  PUBLIC SECTION.\n\n    METHODS
add\n      IMPORTING\n        !p1           TYPE i\n        !p2           TYPE
i\n\n      RETURNING\n        VALUE(result) TYPE i .\n\n  PROTECTED SECTION.\n
PRIVATE SECTION.\n\nENDCLASS.\n\nCLASS zcl_add_test IMPLEMENTATION.\n\n  METHOD
add.\n    result = p1 + p2.\n  ENDMETHOD.\n\nENDCLASS.\n```\n\n## Extra
context\n```abap\n```\n\n"
            }
          ]
        },
        "model": {
          "name": "sap-abap-1",
          "version": "latest",
          "params": {
            "temperature": 0.1,
            "max_tokens": 2000
          }
        }
      }
    }
  }
}'
```

Request Parameter Descriptions

All JSON payloads must be UTF-8 encoded.

Name	Type	Description
`ORCHESTRATION_SERVICE_URL`	string	Orchestration deployment URL (see Get An Orchestration Deployment URL [page 7])
`AUTH_TOKEN`	string	Bearer token generated using the SAP AI Core service key
`config.modules.prompt_templating.prompt.template`	string	For `role:user`, the field `content` holds the actual explanation prompt with the input code (see Prompting Templates [page 183]).
`config.modules.prompt_templating.model.name`	string	Always `sap-abap-1`
`config.modules.prompt_templating.model.name`	string	`latest`
`config.modules.prompt_templating.model.params.temperature`	float	LLM temperature. Good explanations are returned with a value of 0.1, but use 0.0 if you want deterministic results.
`config.modules.prompt_templating.model.params.max_tokens`	integer	Maximum number of output tokens. The model stops predicting when this number is reached. A token is the smallest processing unit of the LLM. On average these LLM tokens are between 2-3 characters.

For more information about input parameters, see Orchestration v2 on the Business Accelerator Hub.

Response Payloads

The response payload, along with the `request_id`, contains two major blocks: `intermediate_results` and `final_result`. `intermediate_results` lists all the intermediate results from the configured Orchestration modules. `final_result` contains the final output after the configured modules have been applied. The final explanation can be found in the `final_result` block in the `content` field.

The response to the example shown above is as follows:

```
{
  "request_id": "a0dd9ce1-c9d2-986c-9c0b-2c93d47b8bc0",
  "intermediate_results": {
    "templating": [
      {
        "role": "user",
        "content": "Explain the ABAP class in the input code below.\nThe
explanation should have the following five sections. Do not provide a header
for the sections.\n- Introduce and summarize the class at the beginning in
```

```
one paragraph.\n- Provide a summary of the variables, without going into too
much detail and provide individual headers. The overall header should be
named `Data Declaration´.\n- If no interfaces are implemented in the class,
skip this section. Provide a summary of the interfaces, without going into
too much detail and provide individual headers. The overall header should be
named `Interfaces´.\n- Provide a summary of each method, without going into
too much detail and provide individual headers. The overall header should
be named `Methods´.\n- Give a brief conclusion that focuses on the usage
of the class. Do not provide an overall header for this section.\nConsider
these guidelines:Focus on dependencies in the code and the side effects
of the codes.Use variable names when needed.\nMention method calls when
needed.\nExplain local includes when needed.\nReject hate speech.\nUse the
following extra context section if provided as additional information to explain
the given code.\n## Input code:\n```abap\nCLASS zcl_add_test DEFINITION\n
PUBLIC\n  FINAL\n  CREATE PUBLIC .\n\n  PUBLIC SECTION.\n\n    METHODS
add\n      IMPORTING\n        !p1           TYPE i\n        !p2           TYPE
i\n\n      RETURNING\n        VALUE(result) TYPE i .\n\n  PROTECTED SECTION.\n
PRIVATE SECTION.\n\nENDCLASS.\n\nCLASS zcl_add_test IMPLEMENTATION.\n\n  METHOD
add.\n    result = p1 + p2.\n  ENDMETHOD.\n\nENDCLASS.\n```\n\n## Extra
context\n```abap\n```\n\n"
      }
    ],
    "input_filtering": {
      "message": "Input Filter passed successfully.",
      "data": {
        "azure_content_safety": {
          "hate": 0,
          "self_harm": 0,
          "sexual": 0,
          "violence": 0,
          "user_prompt_analysis": {
            "attack_detected": false
          }
        }
      }
    },
    "output_filtering": {
      "message": "Choice 0: Output Filter passed successfully.\n",
      "data": {
        "choices": [
          {
            "index": 0,
            "azure_content_safety": {
              "hate": 0,
              "self_harm": 0,
              "sexual": 0,
              "violence": 0
            }
          }
        ]
      }
    },
    "llm": {
      "id": "chatcmpl-f4dbe242d8c84a8280b3d9411e29b460",
      "object": "chat.completion",
      "created": 1765309213,
      "model": "sap-abap-1",
      "choices": [
        {
          "index": 0,
          "message": {
            "role": "assistant",
            "content": "The class `ZCL_ADD_TEST` is a simple utility class
designed to perform addition of two integer values. It contains a single public
method `ADD` that takes two integer parameters and returns their sum. The class
is final and cannot be inherited, ensuring that its behavior remains consistent
and predictable.\n\n**Data Declaration**\n\n**Methods**\n\n**ADD**\nThis method
takes two integer parameters, `P1` and `P2`, and returns their sum. It
```

```
performs a simple addition operation and stores the result in the `RESULT`
variable, which is then returned to the caller.\n\nThe usage of this class is
straightforward. It can be instantiated, and the `ADD` method can be called
with two integer values to get their sum. This class is particularly useful in
scenarios where a simple addition operation needs to be performed repeatedly,
providing a reusable and encapsulated solution."
          },
          "finish_reason": "stop"
        }
      ],
      "usage": {
        "completion_tokens": 172,
        "prompt_tokens": 422,
        "total_tokens": 594
      }
    }
  },
  "final_result": {
    "id": "chatcmpl-f4dbe242d8c84a8280b3d9411e29b460",
    "object": "chat.completion",
    "created": 1765309213,
    "model": "sap-abap-1",
    "choices": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": "The class `ZCL_ADD_TEST` is a simple utility class
designed to perform addition of two integer values. It contains a single public
method `ADD` that takes two integer parameters and returns their sum. The class
is final and cannot be inherited, ensuring that its behavior remains consistent
and predictable.\n\n**Data Declaration**\n\n**Methods**\n\n**ADD**\nThis method
takes two integer parameters, `P1` and `P2`, and returns their sum. It
performs a simple addition operation and stores the result in the `RESULT`
variable, which is then returned to the caller.\n\nThe usage of this class is
straightforward. It can be instantiated, and the `ADD` method can be called
with two integer values to get their sum. This class is particularly useful in
scenarios where a simple addition operation needs to be performed repeatedly,
providing a reusable and encapsulated solution."
        },
        "finish_reason": "stop"
      }
    ],
    "usage": {
      "completion_tokens": 172,
      "prompt_tokens": 422,
      "total_tokens": 594
    }
  }
}
```


Response Parameter Descriptions

Name	Type	Description
`request_id`	string	Generated UUID4 for this request (useful for logging)
`intermediate_results`	object	Intermediate results from orchestration service
`final_result.id`	string	Generated UUID4 for this response

Name	Type	Description
`final_result.object`	string	Prediction object type
`final_result.created`	string	Creation timestamp
`final_result.model`	string	Always `sap-abap-1`
`final_result.choices`	object	List of predictions (typically, there is only one prediction needed and therefore only one element in this list)
`index`	integer	Index of element in prediction list (choices list)
`message.role`	string	Response role
`message.content`	string	Actual prediction/explanation
`message.finish_reason`	string	Applied stopping condition. `stop`: the internal stopping condition from the LLM was reached. `length`: The maximum token length specified in the request was reached.
`final_result.usage.completion_tokens`	integer	Number of prediction tokens (for billing)
`final_result.usage.prompt_tokens`	integer	Number of input tokens (for billing)
`final_result.usage.total_tokens`	integer	Number of input plus output tokens (for billing)

For more information about the input parameters, output parameters, and status codes, see Orchestration v2 on the Business Accelerator Hub.

Related Information

SAP-ABAP-1 [page 175]

1.3.9.2.1.2 Prompting Templates

SAP-ABAP-1 can be used for different code explanation scenarios, for example:

- Explain a complete ABAP class
- Explain an ABAP method
- Explain an ABAP code snippet

For each of these scenarios, you can use one of the following prompt templates and adapt it to your needs. You can also allow additional user input so that users can formulate specific questions about the input code.

To submit these prompt templates to SAP-ABAP-1 via the Orchestration service, insert the prompt template into the user prompt section of your prompt (see Example Payloads for Inferencing: SAP-ABAP-1 [page 177]).

Explain a Complete ABAP Class

You can use the following prompt template to get an explanation of a complete ABAP class. If you only want the definition or implementation section to be explained, include this instruction in the prompt instructions.

```
Explain the ABAP class in the input code below.
The explanation should have the following five sections.
- Introduce and summarize the class at the beginning in one paragraph.
- Provide a summary of the variables, without going into too much detail
and provide individual headers. The overall header should be named `### Data
Declaration´.
- If no interfaces are implemented in the class, skip this section. Provide
a summary of the interfaces, without going into too much detail and provide
individual headers.
The overall header should be named `### Interfaces´.
- Provide a summary of each method, without going into too much detail and
provide individual headers. The overall header should be named `### Methods´.
- Give a brief conclusion that focuses on the usage of the class. Do not provide
an overall header for this section.
Input code:
```abap
{ abap_code_snippet }```
```

Explain an ABAP Method

You can use the following prompt template to get an explanation of an ABAP method. The prompt takes the code for the method as the first input. It also takes the complete class code as additional context to provide the model with more information.

It can be useful to truncate the class and keep only the parts that are relevant to understand the code snippet.

```
Explain the ABAP method in the input code below.
The explanation should have the following four sections.
- Introduce and summarize the code at the beginning in one paragraph.
- Explain the signature of the method.
- Give a step-by-step technical explanation of what each block of the code is
doing and provide individual headers. The overall header should be named '###
Technical Explanation'.
```
```

```
- Give a brief conclusion that focuses on method usage, returning variables
and/or exception handling. Do not provide an overall header for this section.
Input code:
```abap
{ abap_code_snippet }```
Here is the trunctuated form of the class in which the code snippet is part of.
Use this as additional information.
```abap
```

Explain an ABAP Code Snippet

You can use the following prompt template to get an explanation of a code snippet. The prompt takes the code snippet as the first input. It also takes the complete class code as additional context to provide the model with more information.

It can be useful to truncate the class and keep only the parts that are relevant to understand the code snippet.

```
Explain the ABAP input code snippet below.
The explanation should have the following two sections.
- Introduce and summarize the code snippet at the beginning in one paragraph.
- Give a step-by-step technical explanation of what each block of the code is
doing and provide individual headers. The overall header should be named '###
Technical Explanation'
Input code:
```abap
{ abap_code_snippet }```
Here is the trunctuated form of the class in which the code snippet is part of.
Use this as additional information.
```abap
{ abap_class_code }```
```

System Prompt

For security reasons, the system prompt is predefined and managed by SAP. Custom system prompts are not supported.

If a system prompt is included in the request, the request fails and an error is returned.

In typical scenarios, providing a user prompt is sufficient. If additional instructions are required, include them at the beginning or end of the user prompt.

Adding Additional Context on the Input Code

Sometimes, the input code snippet or class alone doesn't provide enough context for the model to generate a good explanation. If you have additional context information about the input code, you can add it to the user prompt. For example, you can add information about the business purpose of the code, the module where the code is used, any other code documentation, more information about important code dependencies of the code, or any other relevant information that helps to understand the code better.

1.3.10 Vertex AI

Context

You can access LLMs through the `foundation-models` and `orchestration` scenarios.

To use a model through the `orchestration` scenario, using the harmonized API, see Orchestration [page 210].

For more information about available models, including conversion rates for tokens, rate limits, and deprecation dates, see SAP Note 3437766.

Accessing Models Through Orchestration

Orchestration offers a harmonized API that allows you to use different models without changing the client code. You likely have an orchestration deployment running in your default resource group. If you want to access models through orchestration in a different resource group, create an orchestration deployment for your chosen resource group. For more information, see .Create a Deployment for Orchestration [page 703].

Access to orchestration of generative AI models is provided under the global AI scenario `orchestration`, which is managed by SAP AI Core.

To access generative AI models using orchestration, add the following information to the `model` module of your orchestration workflow:

- The name of your chosen model
- The version name of your chosen version. If no model version is listed, it is not applicable.

For more information about supported models and associated costs, see SAP Note 3505347

For more information about orchestration workflows, see Orchestration Workflow V2 [page 382].

Accessing Models Through the `foundation-models` Scenario

You can access foundation models by creating a deployment for the model that you want to use. To do so, you need an auth token from your SAP AI Core instance. For more information, see Get an Auth Token [page 6] and Create a Deployment [page 194].

To create your deployment, you'll need the following information:

- The scenario: `foundation-models`
- The `executableId`: `vertexai`
- The name of your chosen model
- The version name of your chosen version. If no model version is listed, it is not applicable.

To use a specific version of a model or to upgrade model versions manually, specify the version of your model deployment. To upgrade automatically, specify the model version as `latest`. For more information, see Model Lifecycle [page 194]. If a model version isn't listed, the model isn't applicable.

Vertex AI models are remote models.

The `executableId` is `gcp-vertexai`

After you've created a deployment for your model, you can consume the model using prompts. To access the model, you need your deployment ID, which you can set as an environment variable.

Check that you've set the following headers:

Header	Value
Authorization	Bearer $AUTH_TOKEN
AI-Resource-Group	The resource group used in the activation steps
$DEPLOYMENT_URL	The deployment URL for your generative AI model. For more information, see Create a Deployment [page 194]. Alternatively, you can replace the `$DEPLOYMENT_URL` placeholder in the curl command with your deployment URL.

Consumption

The following examples show how you can consume various generative AI models using curl. For more information about prompts, see the tutorial Prompt LLMs in the Generative AI Hub in SAP AI Core & Launchpad.

> → Tip
>
> If you use a Windows device, use Windows PowerShell, and replace `curl` with `curl.exe`.

For more information about supported parameters, see Supported Parameters [page 199].

Gemini

For more information from the model provider, see Vertex AI Gemini .

Gemini 2.0 Flash | Gemini 2.0 Flash Lite | Gemini 2.5 Flash | Gemini 2.5 Flash Lite | Gemini 2.5 Pro | Gemini 3.1 Flash Lite | Gemini 3.5 Flash | Gemini-3.1 Flash Image | Gemini 3 Pro Image| Gemini 3.5 Flash Lite | Gemini 3.8 Flash

Text Input

```
curl --request POST --location "$DEPLOYMENT_URL/models/
<modelName>:generateContent" \ #example gemini-1.5-flash:generateContent
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
```

```
--data '{
    "generation_config":{
        "maxOutputTokens":100
    },
    "contents": {
      "role": "user",
      "parts": {
          "text": "Give me a recipe for banana bread."
       }
      },
    "safety_settings": {
      "category": "HARM_CATEGORY_SEXUALLY_EXPLICIT",
      "threshold": "BLOCK_LOW_AND_ABOVE"
    }
}'
```

Image input

```
curl --request POST --location "$DEPLOYMENT_URL/models/
<modelVersion>:generateContent" \ #example gemini-1.5-flash:generateContent
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "contents": {
        "role": "user",
        "parts": [
            {
                "fileData": {
                    "mimeType": "image/png",
                    "fileUri": "filepath/images/scones.jpg"
                }
            },
            {
                "text": "Describe this picture."
            }
        ]
    }
}'
```

Image Generation with Gemini 2.5 Flash

```
curl --request POST --location "$DEPLOYMENT_URL/models/gemini-2.5-flash-
image:generateContent" \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "contents": [
    {
      "role": "user",
      "parts": [
        {
          "text": "A cozy, cinematic coffee shop interior at golden hour,
shallow depth of field, 35mm, soft rim light"
        }
      ]
    }
  ]
}'
```

Gemini Embedding

```
curl --location '$DEPLOYMENT_URL/models/gemini-embedding:predict' \
--header 'AI-Resource-Group: <Resource Group Id>' \
```

```
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "instances": [
    {
      "task_type": "<Task Type>",
      "title": "Behaviour Analysis",
      "content": "What is quid pro quo?"
    }
  ]
}'
```

`Task_type` can take the following values:

- `RETRIEVAL_QUERY`
- `RETRIEVAL_DOCUMENT`
- `SEMANTIC_SIMILARITY`
- `CLASSIFICATION`
- `CLUSTERING`

Gemini Embedding 2

Text

```
curl --location "$DEPLOYMENT_URL/models/gemini-embedding-2:embedContent" \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "content": {
    "parts": [
      {
        "text": "Balance Sheet or Sheets?"
      }
    ]
  }
}'
```

Image

Images must meet the following criteria:

- Compatible Formats: JPEG, PNG, WebP, BMP, HEIC, HEIF, AVIF
- Maximum 6 images per request.
- Maximum 16384 x 16384 pixels per image
- Maximum 1 file per request, of maximum length 6 pages.
  For best quality, use 1 page per PDF.

```
curl --location "$DEPLOYMENT_URL/models/gemini-embedding-2:embedContent" \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "content": {
    "parts": [
      {
        "inlineData": {
          "mimeType": "image/jpeg",
          "data": "<base64-encoded-image>"
        }
      }
```

```
    ]
  }
}'
```

Audio

Audio must meet the following criteria:

- Audio Limits: 180 seconds per request.
- Format: MP3, WAV

Inputs longer than 8192 tokens are silently truncated. All modalities share the total 8192 tokens context window.

```
curl --location "$DEPLOYMENT_URL/models/gemini-embedding-2:embedContent" \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "content": {
    "parts": [
      {
        "inlineData": {
          "mimeType": "audio/mp3",
          "data": "<base64-encoded-audio>"
        }
      }
    ]
  }
}'
```

Video

Videos must meet the following criteria:

- Compatible Formats: MOV, MP4. Codec: AV1, H264, H265, VP9.
- Maximum 1 video per request, of up to 120 frames
- At the default 1 FPS setting the length of the video can be up to 120 seconds. Duration varies when you set different FPS: Duration = Frames/FPS. By default, audio extraction is disabled for video. When the audio track extraction option is enabled, the duration is subject to the maximum audio duration of 180 seconds, and the total video duration is limited by both the audio limit and the token context window (8192 tokens).

Audio must meet the following criteria:

- Audio Limits: 180 seconds per request.
- Format: MP3, WAV

Inputs longer than 8192 tokens are silently truncated. All modalities share the total 8192 tokens context window.

```
curl --location "$DEPLOYMENT_URL/models/gemini-embedding-2:embedContent" \
--header "AI-Resource-Group: $AI_RESOURCE_GROUP" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "content": {
    "parts": [
      {
        "fileData": {
          "mimeType": "video/mp4",
          "fileUri": "gs://<bucket>/<path-to-video>.mp4"
```

```
        }
      }
    ]
  }
}'
```

Removing a Model

If you want to remove a model, delete its deployment. For more information, see Delete a Single Deployment

1.4 Model Consumption Methods

1.4.1 Foundation Models

The foundation models service operates under the global AI scenario `foundation-models`, which is managed by SAP AI Core.

You can access foundation models by creating a deployment for the model that you want to use. To do this, you'll need an auth token from your SAP AI Core instance. For more information, see Get an Auth Token [page 6] and Create a Deployment [page 194].

The following scenarios are available:

Global Scenario	Executable ID	Description
foundation-models	aicore-cohere	Cohere models hosted and accessed through SAP AI Core.
foundation-models	aicore-ibm	Models from IBM that are are hosted and managed by SAP and accessed through SAP AI Core.
foundation-models	aicore-mistralai	Models from Mistral AI that are hosted and managed by SAP and accessed through SAP AI Core.
foundation-models	aicore-nvidia	Models from Nvidia that are hosted and managed by SAP and accessed through SAP AI Core.
foundation-models	aicore-opensource	Open source models that are hosted and managed by SAP and accessed through SAP AI Core.
foundation-models	aws-bedrock	AWS Bedrock provides access to foundation models from Anthropic, Amazon and other providers.
foundation-models	azure-openai	The Azure OpenAI Service provides REST API access to OpenAI's LLMs.
foundation-models	gcp-vertexai	GCP Vertex AI provides access to models from Google.

For more information about available models, including conversion rates for tokens, rate limits and deprecation dates, see SAP Note 3437766.

After creating a deployment for your model, you consume the model using prompts. For more information, see Consume Models [page 198].

1.4.1.1 Get an Auth Token

Authentication token retrieval enables secure access to your SAP instance. Set environment variables and use curl commands to obtain and use an access token. Use this process to authenticate API requests in subsequent steps.

Prerequisites

- curl is installed on your operating system.
- You have the following values from your service key:
  - AI API URL
  - Authentication URL
  - Client ID
  - Client secret

Context

Start by setting the required environment variables, which you can get from your SAP AI Core instance.

Procedure

1. Set up your environment as follows:

```
AI_API_URL=<YOUR AI API URL>
AUTH_URL=<YOUR AUTH URL>
CLIENT_ID=<YOUR CLIENT ID>
CLIENT_SECRET=<YOUR CLIENT SECRET>
RESOURCE_GROUP=default
```

2. Obtain the authentication token by sending the following request:

```
TOKEN=`curl --request POST \
  --url "$AUTH_URL/oauth/token" \
  --header "Content-Type: application/x-www-form-urlencoded" \
  --data "grant_type=client_credentials" \
  --data "client_id=$CLIENT_ID" \
  --data "client_secret=$CLIENT_SECRET"'
```

   The response includes an access token.

```
{"access_token":"ey...", "expires_in":7199, "jti":"...",
"token_type":"bearer"}
```

3. Set the token to use it in the following steps:

```
TOKEN=ey...
```

1.4.1.2 Choose a Model

Decide which LLM you want to deploy and note the following information:

- Executable ID
- Model name
- Model version

> **Note**
>
> - Instead of specifying a model version, specify “latest” to use the latest version of the model available in SAP AI Core.
> - Where the model version isn't listed, it isn't applicable.
>
> For more information, see Model Lifecycle [page 194].

For more information about available models, including conversion rates for tokens, rate limits and deprecation dates, see SAP Note 3437766.

To view the available models, send a GET request to the endpoint: `$AI_API_URL/v2/lm/scenarios/foundation-models/models`.

```
curl --location '$AI_API_URL/v2/lm/scenarios/foundation-models/models' \
--header 'AI-Resource-Group: default' \
--header 'Authorization: Bearer $AUTH_TOKEN'
```

Output Code


```
{
  "count": 41,
  "resources": [
    {
      "accessType": "Remote",
      "allowedScenarios": [
        {
          "executableId": "azure-openai",
          "scenarioId": "foundation-models"
        },
        {
          "executableId": "orchestration",
          "scenarioId": "orchestration"
        }
      ],
      "description": "OpenAI GPT-4o engineered for speed and efficiency,
supporting text, images and audio using Chat Completions API",
      "displayName": "GPT-4o",
      "executableId": "azure-openai",
      "model": "gpt-4o",
      "provider": "OpenAI",
      "versions": [
        {
          "capabilities": [
            "text-generation",
            "image-recognition"
          ],
```

```
            "contextLength": 128000,
            "cost": [
              {
                "inputCost": "0.00312"
              },
              {
                "outputCost": "0.0092"
              }
            ],
            "deprecated": false,
            "inputTypes": [
              "text",
              "image",
              "audio"
            ],
            "isLatest": false,
            "metadata": [
              {
                "meanWinRate": "0.938"
              },
              {
                "chatBotArenaScore": "1285"
              },
              {
                "airBenchRefusalRate": "0.528"
              }
            ],
            "name": "2024-05-13",
            "retirementDate": "",
            "streamingSupported": true
          }, ...
    ]
}
```


The output contains the following parameters, which are needed to create a configuration:

Parameter	Description
count	Number of models available description
description	Description for model
executableId	Executable under which the model is available, required at the time of creating Deployment configuration model
model	Required as modelName at the time of creating Deployment configuration versions
versions	List of available model versions for given model
versions.name	Optional value to provide as modelVersion at the time of creating Deployment Configuration
versions.isLatest	Suggests if the modelVersion is latest. See Model Lifecycle for more information

Next Steps

You make a model available for use by creating a deployment. For more information, see Create a Deployment [page 194].

1.4.1.2.1 Model Lifecycle

Model versions have deprecation dates. Where a model version is specified in a deployment, the deployment stops working on the deprecation date of that model version.

Implement one of the following model upgrade options:

- **Auto Upgrade:** Create a generative AI configuration and deployment or patch a deployment with a new configuration, specifying `modelVersion latest`. When a new `modelVersion` is supported by SAP AI Core, existing generative AI deployments will automatically use the latest version of the given model.
- **Manual Upgrade:** Create a generative AI configuration with your chosen replacement `modelVersion` and use it to patch your deployment. This model version is used in generative AI deployments irrespective of updates to the models supported by SAP AI Core.

> **Note**
>
> If `modelVersion` isn't specified, it will be `latest` by default. To upgrade manually, you **must** specify a `modelVersion`.

1.4.1.3 Create a Deployment

Prerequisites

- You have an SAP AI Core service instance and service key. For more information, see SAP AI Core Initial Setup Documentation.
- You're using the `extended` service plan. For more information, see Service Plans and Update a Service Plan.
- You've chosen the model that you want to make a deployment for. For more information, see Choose a Model [page 192].

Context

You make a model available for use by creating a deployment. You can do so one time for each model and model version. The model deployment includes the `modelName` and `version` of the model you want to access. After the deployment is complete, you have a `deploymentUrl`, which can be used across your organization to access the model version.

Deployments for accessing models are made using the `foundation-models` scenario.

You can view the available models and their details by sending a GET request to the endpoint `{{apiurl}}/v2/lm/scenarios/foundation-models/models`

Check Scenario

Check that you have access to the scenario containing generative AI by sending a GET request to `{{apiurl}}/v2/lm/scenarios`.

> **Note**
>
> You must use the same resource group for all of your generative AI activities. To use a different resource group, these steps must be repeated for each resource group. For more information, see Manage Resource Groups [page 711].

```
curl --location '{{apiurl}}/v2/lm/scenarios' \
--header 'AI-Resource-Group: default' \
--header "Authorization: Bearer $AUTH_TOKEN"
```

Step Result

The scenarios listed contain a scenario with the id `foundation-models`.

Create a Configuration

Create a configuration by sending a POST request to the endpoint `{{apiurl}}/v2/lm/configurations`.

Include details of the model to which you want to provide access by passing in the following parameters:

- `name` is your free choice of identifier.
- `executableId`, `modelName`, and `modelVersion` are provided in SAP Note 3437766.
- `scenarioId` must be `foundation-models`.
- `versionId` is your own version reference.

> **Sample Code**

```
curl --location '$AI_API_URL/v2/lm/configurations' \
--header 'AI-Resource-Group: default' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "name": "yourNameChoice",
    "executableId": "<executableId>",
    "scenarioId": "foundation-models",
    "versionId": "0.0.1",
    "parameterBindings": [
        {
            "key":"modelName",
            "value":"<ModelName>"
        },
        {
            "key": "modelVersion",
            "value": "<ModelVersion>"
        }
  ]
}'
```

→ Tip

You can specify the value `latest` for the `modelVersion` to use the most recent model version available in SAP AI Core.

Step Result

You receive a unique `configurationId` in the response.

Create a Deployment

Create a deployment by sending a POST request to the endpoint `{{apiurl}}/v2/lm/deployments`.

Include the `configurationId` from the previous step in your request.

{·} Sample Code

```
curl --location '$AI_API_URL/v2/lm/deployments' \
--header 'AI-Resource-Group: default' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
 "configurationId": "yourConfigurationId"
}'
```

Get the Deployment URL

Retrieve the details of your deployment by sending a GET request to the endpoint `{{apiurl}}/v2/lm/deployments`.

```
curl --location '$AI_API_URL/v2/lm/deployments/$DEPLOYMENT_ID' \
--header 'AI-Resource-Group: default' \
--header "Authorization: Bearer $AUTH_TOKEN"
```

When the deployment is running, the model can be accessed using the `deploymentUrl` provided in the response. For more information, see Consume Models [page 198].

Related Information

Access the Deployment using the SAP AI SDK for Java

Access the Deployment using the SAP AI SDK for JavaScript

1.4.1.4 Deployment Lifecycle Management

Context

> ⚠ Restriction
>
> Only deployments in a stopped state can be deleted.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Stop a Deployment

Send a PATCH request to the endpoint `$AI_API_URL/v2/lm/deployments/<deploymentid>`. Include the ID of the deployment that you want to stop in your request:

> {·} Sample Code

```
curl PATCH '$AI_API_URL/v2/lm/deployments/<deploymentid>' \
--header 'AI-Resource-Group: default' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
"targetStatus": "STOPPED"
}'
```

Delete a Deployment

Send a DELETE request to the endpoint `$AI_API_URL/v2/lm/deployments/<deploymentid>`. Include the ID of the deployment that you want to stop in your request.

> {·} Sample Code

```
curl DELETE '$AI_API_URL/v2/lm/deployments/<deploymentid>' \
--header 'AI-Resource-Group: default' \
--header "Authorization: Bearer $AUTH_TOKEN" \
```

1.4.1.5 Consume Models

The generative AI hub helps you to complete tasks like summarizing text, inferencing, and transforming content. To do so, you consume a generative AI model by sending a request to the model's endpoint.

Prerequisites

You have the deployment URL for your generative AI model. For more information, see Create a Deployment [page 194].

Context

Ensure that you have the following headers set:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps

> ⚠ Caution
>
> SAP doesn't take any responsibility for the quality of the content in the input to or output of the underlying generative AI models. This includes but isn't limited to bias, hallucinations, or inaccuracies. The user is responsible for verifying the content.

> ⚠ Caution
>
> Don't store personal data in prompts when using the generative AI hub. Personal data is any data that can be used alone, or in combination, to identify the person that the data refers to.

Procedure

Consume your generative AI deployment using your deployment URL and the example payloads provided. For more information, see the following:

- Mistral [page 60]
- NVIDIA [page 67]
- Cohere [page 57]
- Perplexity [page 110]
- OpenAI [page 69]
- Vertex AI [page 185]

- AWS Bedrock [page 52]
- SAP-RPT-1 [page 121]
- SAP-RPT-1.5 [page 133]
- SAP-ABAP-1 [page 175]

1.4.1.5.1 Supported Parameters

The chat completions API supports a range of parameters that influence model responses, influencing aspects like creativity, length, and formatting of the output.

The following parameters are available:

Parameter	Type	Description
messages	array	Standard OpenAI chat completion format
model	string	Model identifier
frequency_penalty	float	Penalize frequent tokens. Range: -2.0 to 2.0
presence_penalty	float	Penalize present tokens. Range: -2.0 to 2.0
logit_bias	map	Modify likelihood of specified tokens
max_tokens	integer	Maximum tokens to generate
max_completion_tokens	integer	Maximum number of tokens that can be generated for a completion, including visible output tokens and reasoning tokens
n	integer	Number of completions to generate
presence_penalty	float	Penalize new tokens based on presence. Range: -2.0 to 2.0
response_format	object	Specify output format
stop	string or array	Stop sequences (up to 4)
stream	boolean	Enable streaming responses
temperature	float	Sampling temperature. Range: 0.0 to 2.0
top_logprobs	integer	The number of most likely tokens to return at each token position, with corresponding log probability. Logprobs must be set to true if this parameter is used. Range: 0 to 20
logprobs	object	Whether to return log probabilities of the output tokens or not. True returns the log probabilities of each output token

Parameter	Type	Description
top_p	float	Nucleus sampling parameter. Range: 0.0 to 1.0
tools	array	List of tools the model may call
tool_choice	string or object	Control tool calling strategy

1.4.1.6 Foundation Model Tutorials

We provide tutorials that demonstrate use cases for accessing generative AI models in the generative AI hub using foundation scenarios. To access each model version, you'll need to create a separate deployment.

For a complete list of tutorials on foundation models, see Generative AI with SAP AI Core - Foundation Models.

Related Information

Libraries and SDKs

1.4.2 Batch Consumption

Process large volumes of LLM requests asynchronously in SAP AI Core. Submit a collection of requests as a single input file and the service processes them in the background, then writes the results to your object store.

> **Note**
>
> Batch consumption only supports native LLM calls. Orchestration requests aren't supported.
>
> Batch consumption is available in EU and US regions, except `prod-euonly` and sovereign cloud deployments.

For more information about available models, including conversion rates for tokens, rate limits, and deprecation dates, see SAP Note 3437766.

Key Capabilities

- Process hundreds or thousands of LLM requests in a single submission
- Reduced cost compared to synchronous inference calls
- Automatic retry handling for transient provider errors

- No rate limit management required on the client side

Prerequisites

- You have a valid SAP AI Core service instance with access to the generative AI hub. For more information, see Use a Service Key.
- You have an object store secret for one of the following supported providers is registered in SAP AI Core. For more information, see Register Your Object Store Secret.
  - Amazon S3
  - Azure Blob Storage
  - Google Cloud Storage
  - Alibaba Cloud OSS
  - SAP HANA Cloud, Data Lake

Workflow

- Prepare and upload your input file. For more information, see Prepare and Upload Your Input File [page 201].
- Create a batch job that the system processes asynchronously in the background. For more information, see Create a Batch Job [page 203].
- Check the status of your batch job. For more information, see Check Batch Status [page 205].
- Retrieve your results from your object store. For more information, see Retrieve Results [page 207].
- Get the details of your batch. For more information, see List Batch Jobs [page 208] and Get Batch Details [page 208].
- Manage the lifecycle of your batch. For more information, see Cancel a Batch Job [page 209] and Delete a Batch Job [page 210].

1.4.2.1 Prepare and Upload Your Input File

Prerequisites

- You have an object store secret for one of the following providers. Other providers are not supported for batch consumption. For more information, see Register Your Object Store Secret.
  - Amazon S3
  - Azure Blob Storage
  - Google Cloud Storage

- Alibaba Cloud OSS
- SAP HANA Cloud, Data Lake

Prepare Your Input File

Batch jobs require a JSON Lines file as input.

Each line must be a self-contained, valid JSON object representing one inference request, and lines must not be separated by commas.

For detailed input file specifications, see the Azure OpenAI Batch documentation .

The following fields are required per line:

Field	Description
`custom_id`	A unique identifier for the request. Used to match responses to inputs.
`method`	The HTTP method. The following methods are supported: • `POST`
`url`	The inference endpoint path. The following paths are supported: • `/v1/chat/completions`
`body`	The request body. The request body must include the following: • `model` • `messages` Optionally, the request body can include parameters.

> **Note**
>
> All requests in the input file must use the same model.

Example

```
{"custom_id": "request-1", "method": "POST", "url": "/v1/chat/completions",
"body": {"model": "gpt-4.1", "messages": [{"role": "user", "content": "What is
machine learning?"}], "max_tokens": 150}}
{"custom_id": "request-2", "method": "POST", "url": "/v1/chat/completions",
"body": {"model": "gpt-4.1", "messages": [{"role": "user", "content": "Explain
neural networks in simple terms"}], "max_tokens": 150}}
```

Upload Your Input File

Upload the JSONL input file to your object store. The batch service accesses it using the `ai://` URI scheme, which references a registered object store secret.

The URI format is:

```
ai://<object_store_secret_name>/<file_path>/<file_name>.jsonl
```

Where `<object_store_secret_name>` is the name of the object store secret registered in SAP AI Core.

> **Note**
>
> You will need your full URI when creating the batch job.

For instructions on uploading files to your specific object store, see the documentation from the storage provider.

1.4.2.2 Create a Batch Job

Prerequisites

You've prepared and uploaded your batch input file. For more information, see Prepare and Upload Your Input File [page 201].

Procedure

Submit a batch job by sending a POST request to the endpoint `$AI_API/llm-batch-service/v1/batches`.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Include the following in your request:

Parameter	Type	Description
`type`	string	Batch processing type. The following values are supported: • `llm-native`
`input.uri`	string	URI of the input file in the object store, The following file types are supported: • `.jsonl`
`output.uri`	string	URI of the output directory in the object store. Must end with `/`.
`spec.provider`	string	LLM provider. The following providers are supported: • `azure-openai`
`spec.model`	string	Model name. **Note** The model must match the value specified in the input file.

Sample Code

```
curl -X POST "$AI_API/llm-batch-service/v1/batches" \
    -H 'AI-Resource-Group: <resource_group>' \
    -H 'Content-Type: application/json' \
    -H "Authorization: Bearer $AUTH_TOKEN" \
    -D '{
        "type": "llm-native",
        "input": {
            "uri": "ai://<object_store_secret_name>/<input_file>.jsonl"
        },
        "output": {
            "uri": "ai://<object_store_secret_name>/<output_folder>/"
        },
        "spec": {
            "provider": "azure-openai",
            "model": "gpt-4.1"
        }
    }'
```

Results

The service schedules the job for processing and returns a unique batch ID.

Output Code

```
{
  "id": "550e8400-e29b-41d4-a716-446655440000",
  "created_at": "2026-01-07T10:30:00Z",
  "status": "PENDING",
  "message": "Batch job scheduled"
```

```
}
```

Next Steps

Use your batch ID to check the status of your batch, and retrieve your results. For more information, see Check Batch Status [page 205] and Retrieve Results [page 207].

1.4.2.3 Check Batch Status

Prerequisites

You've created a batch job, and know its ID. For more information, see Create a Batch Job [page 203].

Procedure

Batches are processed asynchronously in the background. You can monitor the progress of your batch by sending a GET request to the endpoint: `$AI_API/llm-batch-service/v1/batches/$BATCH_ID/status`. Include your batch ID in your request.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Sample Code

```
curl -X GET "$AI_API/llm-batch-service/v1/batches/$BATCH_ID/status" \
    -H 'AI-Resource-Group: <resource_group>' \
    -H 'Content-Type: application/json' \
    -H "Authorization: Bearer $AUTH_TOKEN" \
```

Results

The response includes the current status of your batch.

Output Code

```
{
  "current_status": "IN_PROGRESS",
  "target_status": "COMPLETED",
  "updated_at": "2026-01-07T10:35:00Z",
  "message": "Processing batch requests"
}
```

Statuses include the following:

Status	Description	Action
`PENDING`	Job is queued and waiting to start.	Continue to poll the job status.
`PREPARING_INPUT`	The system validates and prepares the input file for the LLM provider.	Continue to poll the job status.
`INPUT_PREPARED`	The input file is ready. The system will submit the job to the provider.	Continue to poll the job status.
`IN_PROGRESS`	The LLM provider is processing the job.	Continue to poll the job status.
`PREPARING_OUTPUT`	Processing is complete. The system writes output files to the object store.	Continue to poll the job status.
`COMPLETED`	The job finished successfully. Output files are available.	Retrieve the results from the object store.
`FAILED`	The job failed due to an error.	Check the `message` field and retry if appropriate.
`CANCELLING`	The system is cancelling the job and cleaning up resources.	Continue to poll the job status.
`CANCELLED`	Job was cancelled by the user.	No action is required.

Next Steps

When your batch is `COMPLETED`, you can view the output in your object store. For more information, see Retrieve Results [page 207].

1.4.2.4 Retrieve Results

Prerequisites

- You've created a batch job. For more information, see Create a Batch Job [page 203].
- The status of your batch job is `COMPLETED`. For more information, see Check Batch Status [page 205].

Context

The batch service writes output to a subdirectory named after the batch ID, under the output URI specified at job creation.

> **Example**
>
> If the output URI was `ai://my-object-store/output/`, the results are written to:
>
> ```
> ai://my-object-store/output/<batch_id>/output.jsonl
> ```

> **Example**
>
> If any individual requests failed during processing, an error file is written alongside the output file:
>
> ```
> ai://my-object-store/output/<batch_id>/error.jsonl
> ```

Output File Format

Each line in the output file corresponds to one request from the input file, matched by `custom_id`:

```
{"id": "batch-response-1", "custom_id": "request-1", "response": {"status_code":
200, "body": {"id": "chatcmpl-123", "choices": [{"message": {"content": "Machine
learning is a subfield of..."}}]}}}
{"id": "batch-response-2", "custom_id": "request-2", "response": {"status_code":
200, "body": {"id": "chatcmpl-124", "choices": [{"message": {"content": "Neural
networks are computational models..."}}]}}}
```

> **Note**
>
> The status `COMPLETED` indicates that the batch as a whole was processed. Individual requests within the batch may still have failed. Check `response.status_code` for each entry in the output file, and review the error file for failed requests.

1.4.2.5 Get Batch Details

You can retrieve the full details of a batch job, including input and output URIs, provider, model, and current status by sending a GET request to the endpoint:`$AI_API/llm-batch-service/v1/batches/$BATCH_ID`. Include the batch ID in your request.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Sample Code

```
curl -X GET "$AI_API/llm-batch-service/v1/batches/$BATCH_ID" \
    -H 'AI-Resource-Group: <resource_group>' \
    -H 'Content-Type: application/json' \
    -H "Authorization: Bearer $AUTH_TOKEN" \
```

Output Code

```
{
  "id": "550e8400-e29b-41d4-a716-446655440000",
  "type": "llm-native",
  "provider": "azure-openai",
  "created_at": "2026-01-07T10:30:00Z",
  "input": {
    "uri": "ai://my-object-store/input/input-batch.jsonl"
  },
  "output": {
    "uri": "ai://my-object-store/output/"
  },
  "spec": {
    "model": "gpt-4.1"
  },
  "status": {
    "current_status": "COMPLETED",
    "target_status": "COMPLETED",
    "updated_at": "2026-01-07T10:45:00Z",
    "message": "Batch job completed successfully"
  }
}
```

1.4.2.6 List Batch Jobs

To list all batch jobs for the current tenant and resource group, send a GET request to the endpoint: `$AI_API/llm-batch-service/v1/batches`.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Sample Code

```
curl -X GET "$AI_API/llm-batch-service/v1/batches" \
    -H 'AI-Resource-Group: <resource_group>' \
    -H 'Content-Type: application/json' \
    -H "Authorization: Bearer $AUTH_TOKEN" \
```

Output Code

```
{
  "count": 2,
  "resources": [
    {
      "id": "550e8400-e29b-41d4-a716-446655440000",
      "type": "llm-native",
      "provider": "azure-openai",
      "created_at": "2026-01-07T10:30:00Z",
      "status": "COMPLETED"
    },
    {
      "id": "661f9511-f3ac-52e5-b827-557766551111",
      "type": "llm-native",
      "provider": "azure-openai",
      "created_at": "2026-01-07T11:00:00Z",
      "status": "IN_PROGRESS"
    }
  ]
}
```

1.4.2.7 Cancel a Batch Job

To cancel a batch job that is in a non-terminal state, for example `PENDING` or `IN_PROGRESS`, send a PATCH request to the endpoint: `$AI_API/llm-batch-service/v1/batches/$BATCH_ID/cancel`. Include your batch ID in your request.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Sample Code

```
curl -X PATCH "$AI_API/llm-batch-service/v1/batches/$BATCH_ID/cancel" \
    -H 'AI-Resource-Group: <resource_group>' \
    -H 'Content-Type: application/json' \
    -H "Authorization: Bearer $AUTH_TOKEN" \
```

Note

Cancellation is asynchronous. The job transitions to `CANCELLED` status after any in-flight provider requests have been terminated. To check the status of your cancellation, see Check Batch Status [page 205].

1.4.2.8 Delete a Batch Job

Deleting a job removes its metadata from the service. Output files in the object store are not affected.

Restriction

Only batch jobs in `COMPLETED`, `FAILED`, or `CANCELLED` state can be deleted. To check the status of your cancellation, see Check Batch Status [page 205].

To delete a batch job and its associated metadata, send a DELETE request to the endpoint: `$AI_API/llm-batch-service/v1/batches/$BATCH_ID`. Include your batch ID in your request.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

```
curl -X DELETE "$$AI_API/llm-batch-service/v1/batches/$BATCH_ID" \
    -H 'AI-Resource-Group: <resource_group>' \
    -H 'Content-Type: application/json' \
    -H "Authorization: Bearer $AUTH_TOKEN" \
```

1.4.3 Orchestration

The orchestration service runs on SAP AI Core under the global AI scenario `orchestration`. It provides unified access to multiple generative AI models through consistent code, configuration, and deployment.

Orchestration offers a harmonized API that allows you to use different foundation models without changing the client code. To use different foundation models and versions, create at least one orchestration deployment or use the orchestration deployment in your default resource group.

Key features of orchestration include:

- **Templating:** This feature lets you compose prompts with placeholders that are populated during inference.
- **Content filtering:** This feature allows you to restrict the type of content passed to and received from generative AI models.
- **Data masking:** This feature enables anonymization or pseudonymization of data before passing it to a generative AI model. With pseudonymization, masked data in the model's response is automatically restored.
- **Grounding:** This feature lets you integrate external, domain-specific, or real-time data to enhance pretrained models with contextually relevant information beyond their general training material.
- **Translation:** This feature lets you add translation capabilities for both input and output in your orchestration workflow.

The following scenarios are available:

Global Scenario	Executable ID	Description
`orchestration`	`orchestration`	Access to orchestration of generative AI models is provided under the global AI scenario `orchestration`, which is managed by SAP AI Core.

> → Tip
>
> To integrate orchestration into your application, the SAP Cloud SDK for AI is recommended. For more information, see Libraries and SDKs.

1.4.3.1 Orchestration Workflow V1 (Deprecated)

In a basic orchestration scenario, you can combine different modules from orchestration into a pipeline that can be executed with a single API call. Within the pipeline, the response from one module is used as the input for the next module.

> ⓘ Note
>
> This initial orchestration endpoint is deprecated and is scheduled for decommissioning on December 31, 2026. Following that date, the endpoint will no longer be available. We recommend that you create new orchestration workflows using version 2, and that you migrate existing workflows from version 1 to version 2. To use version 2, you'll need to update your endpoint from `/completion` to `/v2/completion` and modify your existing payloads. For more information, see Orchestration Workflow V2 [page 382].

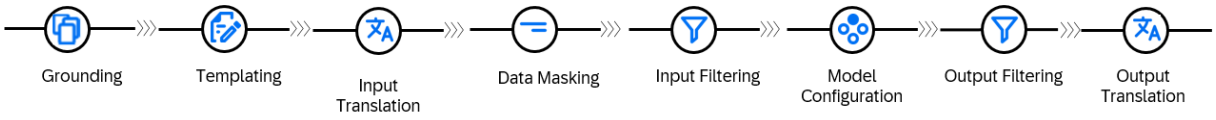
Prerequisites

You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].

Workflow

The order in which the pipeline is executed is defined centrally in orchestration. However, you can configure the details for each module and omit optional modules by passing an orchestration configuration in JSON format with the request body.

The image below is interactive. Click each part of the workflow for more information.



- Grounding [page 392]
- Templating [page 344]
- Input Translation [page 358]
- Data Masking [page 360]
- Input Filtering [page 370]
- Model Configuration [page 374]
- Output Filtering [page 375]
- Output Translation [page 380]

Using Orchestration

To interact with the orchestration service, send a POST request to your **Orchestration URL**. Ensure that your request includes the appropriate headers and a JSON payload that adheres to the specified schema.

Base URL

The base URL is `{{Orchestration URL}}`, and is the URL of your RUNNING orchestration deployment.

Endpoint

The endpoint for orchestration v1 is POST `/completion`.

Include the following headers in your request:

- Authorization: Bearer <your-access-token>
- content-type: application/json
- AI-Resource-Group: <your-resource-group>

Request Body

Your request body should follow the following structure:

```
{
    "orchestration_config": {
    "module_configurations": {
      "llm_module_config": { ... }, // required module
```

```
      "templating_module_config": { ...}, // required module
      "<optional module config>": { ...}, // optional modules such as grounding,
content filtering etc.
      ...
    }
  },
  "input_params": { ... }
}
```

The templating and model configuration modules are mandatory.

All other modules are optional, and you can choose which to include depending on your use case.

For more information, see the following:

Orchestration Module	Config	Documentation Link
Grounding	`grounding_module_config`	Grounding [page 392]
Templating (mandatory)	`templating_module_config`	Templating [page 344]
Model Config (mandatory)	`llm_module_config`	Model Configuration [page 374]
Content Filtering	`filtering_module_config`	Input Filtering [page 370] Output Filtering [page 375]
Data Masking	`masking_module_config`	Data Masking [page 360]
Input Translation	`input_translation_module_config`	Input Translation [page 358]
Output Translation	`output_translation_module_config`	Output Translation [page 380]

Example

This is a sample orchestration request which includes all the above mentioned orchestration modules. In each module section you can find the details of its configuration.

Populate the following variables with your own values:

Variable	Description
ORCH_DEPLOYMENT_URL	Orchestration Deployment URL
TOKEN	Bearer token generated using SAP AI Core service key
RESOURCE_GROUP	The ID of the AI resource group that contains your orchestration deployment
data_repositories	Data repository IDs of repositories to be used for document grounding

Sample Code

```
curl --request POST \
```

```
  --url $ORCH_DEPLOYMENT_URL/completion \
  --header "Authorization: Bearer $TOKEN"  \
  --header "AI-Resource-Group: $RESOURCE_GROUP"  \
  --header 'content-type: application/json' \
  --data '{
  "orchestration_config": {
    "module_configurations": {
      "grounding_module_config": {
        "type": "document_grounding_service",
        "config": {
          "filters": [
            {
              "id": "filter1",
              "data_repositories": [
                "<placeholder for data repository ID>"
              ],
              "search_config": {
                "max_chunk_count": 5
              },
              "data_repository_type": "vector"
            }
          ],
          "input_params": ["groundingRequest"],
          "output_param": "groundingOutput"
        }
      },
      "llm_module_config": {
        "model_name": "gpt-4o",
        "model_params": {},
        "model_version": "latest"
      },
      "templating_module_config": {
        "template": [
          {
            "role": "user",
            "content": "You are a precise and reliable assistant. Using only
the provided context, generate a concise and accurate summary relevant to
the request. Do not infer or generate information beyond the given context.
If the requested information is not available in the context, clearly state
that. Request: {{ ?groundingRequest }} Context: {{ ?groundingOutput }}"
          }
        ],
        "defaults": {}
      },
      "filtering_module_config": {
        "input": {
          "filters": [
            {
              "type": "azure_content_safety",
              "config": {
                "Hate": 2,
                "SelfHarm": 2,
                "Sexual": 2,
                "Violence": 2
              }
            },
            {
              "type": "llama_guard_3_8b",
              "config": {
                "violent_crimes": true,
                "non_violent_crimes": true,
                "sex_crimes": true,
                "child_exploitation": false,
                "defamation": false,
                "specialized_advice": false,
                "privacy": false,
                "intellectual_property": false,
                "indiscriminate_weapons": false,
```

```
                "hate": false,
                "self_harm": false,
                "sexual_content": false,
                "elections": false,
                "code_interpreter_abuse": false
              }
            }
          ]
        },
        "output": {
          "filters": [
            {
              "type": "azure_content_safety",
              "config": {
                "Hate": 2,
                "SelfHarm": 2,
                "Sexual": 2,
                "Violence": 2
              }
            }
          ]
        }
      },
      "masking_module_config": {
        "masking_providers": [
          {
            "type": "sap_data_privacy_integration",
            "method": "anonymization",
            "entities": [
              { "type": "profile-email" },
              { "type": "profile-person" },
              { "type": "profile-phone" },
              { "type": "profile-org" },
              { "type": "profile-location" }
            ]
          }
        ]
      },
      "input_translation_module_config": {
        "type": "sap_document_translation",
        "config": {
          "source_language": "de-DE",
          "target_language": "en-US"
        }
      },
      "output_translation_module_config": {
        "type": "sap_document_translation",
        "config": {
          "target_language": "de-DE"
        }
      }
    }
  },
  "input_params": {
    "groundingRequest": "Was sind die Schritte zur Ersteinrichtung im SAP AI
Core?"
  }
}'
```

Output Code

```
{
  "request_id": "2fa2fd2a-8fb9-9e36-94b8-de17fd1f3e17",
  "module_results": {
    "grounding": {
      "message": "Grounding module executed successfully on grounding input.",
```

```
      "data": {
        "grounding_query": "grounding call",
        "grounding_result": "5 Initial Setup\nY ou provision SAP AI Core
from the SAP BTP cockpit in SAP Business T echnology Platform. After
provisioning,\nyou will have your service key, which provides URLs and
credentials for accessing the SAP AI Core instance\nPrerequisites\n our SAP
BTP administrator has access to a global account on SAP Business T echnology
Platform. For\nmore information, see Getting a Global Account .\n our SAP
BTP administrator has set the entitlement to a subaccount.\nContext\nThe
SAP AI Core service is a tenant-aware reuse service. It isolates tenants
based on the ID of the zone\n(representing the subaccount). The SAP AI Core
service instance is created within a subaccount. Each\nsubaccount represents
an SAP AI Core tenant.\n  Note\nThe SAP AI Core service does not isolate
tenants based on the service instance ID. If you create multiple\nservice
instances within the same subaccount, all of them will reference the same
SAP AI Core tenant.\n48 PUBLICSAP AI Core\nInitial Setup```The steps below
guide you through the provisioning procedure, Alternatively, a booster is
available for both\nSAP AI Core and SAP AI Launchpad. For more information,
see AI Boosters T utorial\n . If you choose to use the\nbooster, you can
skip the remaining steps to SAP AI Core Starter T utorials [page 67] .\n5.1
Enabling the Service in Cloud Foundry\nEnable SAP AI Core using the standard
procedures for the SAP BTP Cloud Foundry environment.\nY ou provision SAP AI
Core from the SAP BTP cockpit in SAP Business T echnology Platform. After
you have\nprovisioned the service, you will have your service key, which
provides URLs and credentials for accessing the\nSAP AI Core instance.\nThe
SAP AI Core service is a tenant-aware reuse service. It isolates tenants
based on the ID of the zone\n(representing the subaccount). The SAP AI Core
service instance is created within a subaccount. Each\nsubaccount represents
an SAP AI Core tenant.\n  Note```and register your object store for training
data and trained models. Productize your\nAI content and expose it as a
service to consumers in the SAP BTP marketplace.\nGenerative AI hubChoose
from a selection of generative AI models for prompt experimentation,
and\nprompt lifecycle management.\nProcess Flow Between SAP AI Core and
SAP AI Launchpad\nSAP AI Core\nWhat Is SAP AI Core? PUBLIC 7```5.2
Enabling the Service in the Kyma Environment\nEnable SAP AI Core using the
standard procedures for the SAP BTP Kyma environment.\nProcedure\n1. Create
a service instance in the Kyma environment.\n2. Y ou can then bind the
service instance to your application, or you can create a service key to
communicate\ndirectly with the service instance. See Using SAP BTP Services
in the Kyma Environment .\nRelated Information\nUsing SAP BTP Services in the
Kyma Environment\n66 PUBLICSAP AI Core\nInitial Setup"
      }
    },
    "templating": [
      {
        "content": "You are a precise and reliable assistant. Using only
the provided context, generate a concise and accurate summary relevant to
the request. Do not infer or generate information beyond the given context.
If the requested information is not available in the context, clearly state
that. Request: Was sind die Schritte zur Ersteinrichtung im SAP AI Core?
Context: 5 Initial Setup\nY ou provision SAP AI Core from the SAP BTP
cockpit in SAP Business T echnology Platform. After provisioning,\nyou will
have your service key, which provides URLs and credentials for accessing
the SAP AI Core instance\nPrerequisites\n our SAP BTP administrator has
access to a global account on SAP Business T echnology Platform. For\nmore
information, see Getting a Global Account .\n our SAP BTP administrator
has set the entitlement to a subaccount.\nContext\nThe SAP AI Core service
is a tenant-aware reuse service. It isolates tenants based on the ID of
the zone\n(representing the subaccount). The SAP AI Core service instance
is created within a subaccount. Each\nsubaccount represents an SAP AI Core
tenant.\n  Note\nThe SAP AI Core service does not isolate tenants based on
the service instance ID. If you create multiple\nservice instances within
the same subaccount, all of them will reference the same SAP AI Core
tenant.\n48 PUBLICSAP AI Core\nInitial Setup```The steps below guide you
through the provisioning procedure, Alternatively, a booster is available
for both\nSAP AI Core and SAP AI Launchpad. For more information, see AI
Boosters T utorial\n . If you choose to use the\nbooster, you can skip
the remaining steps to SAP AI Core Starter T utorials [page 67] .\n5.1
```

```
Enabling the Service in Cloud Foundry\nEnable SAP AI Core using the standard
procedures for the SAP BTP Cloud Foundry environment.\nY ou provision SAP AI
Core from the SAP BTP cockpit in SAP Business T echnology Platform. After
you have\nprovisioned the service, you will have your service key, which
provides URLs and credentials for accessing the\nSAP AI Core instance.\nThe
SAP AI Core service is a tenant-aware reuse service. It isolates tenants
based on the ID of the zone\n(representing the subaccount). The SAP AI Core
service instance is created within a subaccount. Each\nsubaccount represents
an SAP AI Core tenant.\n  Note```and register your object store for training
data and trained models. Productize your\nAI content and expose it as a
service to consumers in the SAP BTP marketplace.\nGenerative AI hubChoose
from a selection of generative AI models for prompt experimentation,
and\nprompt lifecycle management.\nProcess Flow Between SAP AI Core and
SAP AI Launchpad\nSAP AI Core\nWhat Is SAP AI Core? PUBLIC 7```5.2
Enabling the Service in the Kyma Environment\nEnable SAP AI Core using the
standard procedures for the SAP BTP Kyma environment.\nProcedure\n1. Create
a service instance in the Kyma environment.\n2. Y ou can then bind the
service instance to your application, or you can create a service key to
communicate\ndirectly with the service instance. See Using SAP BTP Services
in the Kyma Environment .\nRelated Information\nUsing SAP BTP Services in the
Kyma Environment\n66 PUBLICSAP AI Core\nInitial Setup",
        "role": "user"
      }
    ],
    "input_translation": {
      "message": "Translation module executed successfully on messages with
roles: ['user'].",
      "data": {
        "translated_template": "[{\"content\": \"You are a precise and
reliable assistant. Using only the provided context, generate a concise
and accurate summary relevant to the request. Do not infer or generate
information beyond the given context. If the requested information is not
available in the context, clearly state that. Request: What are the initial
setup steps in the SAP AI Core? Context: 5 Initial Setup\\nY ou commission
SAP AI Core from the SAP BTP cockpit in SAP Business T echnology Platform.
After provisioning,\\nyou want to have your service key, which provides URLs
and credentials for accessing the SAP AI Core instance\\nPrerequisites\\n
our SAP BTP administrator has access to a global account on SAP Business T
echnology Platform. For\\nmore information, see Getting a Global Account.\\n
our SAP BTP administrator has set the entitlement to a subaccount.\\nContext\
\nThe SAP AI Core service is a tenant-aware reuse service. It isolates
tenants based on the ID of the zone\\n(representing the subaccount). The SAP
AI Core service instance is created within a subaccount. Each\\nsubaccount
represents to SAP AI Core tenant.\\n Note\\nThe SAP AI Core service does
not isolate tenants based on the service instance ID. If you create multiple\
\nservice instances within the same subaccount, all of them will reference
the same SAP AI Core tenant.\\n48 PUBLICSAP AI Core\\nInitial Setup```The
steps below guide you through the provisioning procedure, Alternatively,
a booster is available for both\\nSAP AI Core and SAP AI Launchpad. For
more information, see AI Boosters T utorial\\n. If you choose to use the\
\nbooster, you can skip the remaining steps to SAP AI Core Starter T utorials
[page 67].\\n5.1 Enabling the Service in Cloud Foundry\\nEnable SAP AI Core
using the standard procedures for the SAP BTP Cloud Foundry environment.\
\nY ou provision SAP AI Core from the SAP BTP cockpit in SAP Business T
echnology Platform. After you have\\nprovisioned the service, you want to
have your service key, which provides URLs and credentials for accessing the\
\nSAP AI Core instance.\\nThe SAP AI Core service is a tenant-aware reuse
service. It isolates tenants based on the ID of the zone\\n(representing the
subaccount). The SAP AI Core service instance is created within a subaccount.
Each\\nsubaccount represents to SAP AI Core tenant.\\n Note```and register
your object store for training data and trained models. Productize your\\nAI
content and expose it as a service to consumers in the SAP BTP marketplace.\
\nGenerative AI hubChoose from a selection of generative AI models for prompt
experimentation, and\\nprompt lifecycle management.\\nProcess Flow Between
SAP AI Core and SAP AI Launchpad\\nSAP AI Core\\nWhat Is SAP AI Core? PUBLIC
7```5.2 Enabling the Service in the Kyma Environment\\nEnable SAP AI Core
using the standard procedures for the SAP BTP Kyma environment.\\nProcedure\
\n1. Create a service instance in the Kyma environment.\\n2. Y ou can then
```

```
bind the service instance to your application, or you can create a service
key to communicate\\ndirectly with the service instance. See Using SAP
BTP Services in the Kyma Environment.\\nRelated Information\\nUsing SAP BTP
Services in the Kyma Environment\\n66 PUBLICSAP AI Core\\nInitial Setup\",
\"role\": \"user\"}]"
      }
    },
    "input_masking": {
      "message": "Masking module executed successfully on LLM input.",
      "data": {
        "masked_template": "[{\"content\": \"You are a precise and reliable
assistant. Using only the provided context, generate a concise and accurate
summary relevant to the request. Do not infer or generate information beyond
the given context. If the requested information is not available in the
context, clearly state that. Request: What are the initial setup steps
in the MASKED_ORG? Context: 5 Initial Setup\\nY ou commission MASKED_ORG
from the MASKED_ORG cockpit in MASKED_ORG. After provisioning,\\nyou want
to have your service key, which provides URLs and credentials for accessing
the MASKED_ORG instance\\nPrerequisites\\n our MASKED_ORG administrator has
access to a global account on MASKED_ORG. For\\nmore information, see Getting
a Global Account.\\n our MASKED_ORG administrator has set the entitlement
to a subaccount.\\nContext\\nThe MASKED_ORG service is a tenant-aware reuse
service. It isolates tenants based on the ID of the zone\\n(representing the
subaccount). The MASKED_ORG service instance is created within a subaccount.
Each\\nsubaccount represents to MASKED_ORG tenant.\\n Note\\nThe MASKED_ORG
service does not isolate tenants based on the service instance ID. If
you create multiple\\nservice instances within the same subaccount, all
of them will reference the same MASKED_ORG tenant.\\n48 MASKED_ORG AI
Core\\nInitial Setup```The steps below guide you through the provisioning
procedure, Alternatively, a booster is available for both\\nMASKED_ORG and
MASKED_ORG Launchpad. For more information, see AI Boosters T utorial\\n.
If you choose to use the\\nbooster, you can skip the remaining steps
to MASKED_ORG Starter T utorials [page 67].\\n5.1 Enabling the Service
in Cloud Foundry\\nEnable MASKED_ORG using the standard procedures for
the MASKED_ORG Cloud Foundry environment.\\nY ou provision MASKED_ORG from
the MASKED_ORG cockpit in MASKED_ORG. After you have\\nprovisioned the
service, you want to have your service key, which provides URLs and
credentials for accessing the\\nMASKED_ORG instance.\\nThe MASKED_ORG service
is a tenant-aware reuse service. It isolates tenants based on the ID of
the zone\\n(representing the subaccount). The MASKED_ORG service instance
is created within a subaccount. Each\\nsubaccount represents to MASKED_ORG
tenant.\\n Note```and register your object store for training data and
trained models. Productize your\\nAI content and expose it as a service to
consumers in the MASKED_ORG marketplace.\\nGenerative AI hubChoose from a
selection of generative AI models for prompt experimentation, and\\nprompt
lifecycle management.\\nProcess Flow Between MASKED_ORG and MASKED_ORG
Launchpad\\nMASKED_ORG\\nWhat Is MASKED_ORG? PUBLIC 7```5.2 Enabling the
Service in the MASKED_PERSON Environment\\nEnable MASKED_ORG using the
standard procedures for the MASKED_ORG MASKED_PERSON environment.\\nProcedure\
\n1. Create a service instance in the MASKED_PERSON environment.\\n2. Y ou
can then bind the service instance to your application, or you can create a
service key to communicate\\ndirectly with the service instance. See Using
MASKED_ORG in the MASKED_PERSON Environment.\\nRelated Information\\nUsing
MASKED_ORG in the MASKED_PERSON Environment\\n66 MASKED_ORG AI Core\\nInitial
Setup\", \"role\": \"user\"}]"
      }
    },
    "input_filtering": {
      "message": "Filtering module executed successfully on content: no
violations detected.",
      "data": {
        "azure_content_safety": {
          "Hate": 0,
          "SelfHarm": 0,
          "Sexual": 0,
          "Violence": 0
        },
        "llama_guard_3_8b": {
```

```
          "violent_crimes": false,
          "non_violent_crimes": false,
          "sex_crimes": false
        }
      }
    },
    "llm": {
      "id": "chatcmpl-BZu6RM8Z1wf74sgGu8gbtwxnoSETM",
      "object": "chat.completion",
      "created": 1747896839,
      "model": "gpt-4o-2024-08-06",
      "system_fingerprint": "fp_ee1d74bde0",
      "choices": [
        {
          "index": 0,
          "message": {
            "role": "assistant",
            "content": "The initial setup steps in MASKED_ORG involve:\n\n1.
Enabling the service in Cloud Foundry by using standard procedures.\n2.
Provisioning MASKED_ORG from the MASKED_ORG cockpit.\n3. Creating a service
instance in the MASKED_PERSON environment.\n4. Binding the service instance
to your application or creating a service key for direct communication with
the service instance."
          },
          "finish_reason": "stop"
        }
      ],
      "usage": {
        "completion_tokens": 78,
        "prompt_tokens": 722,
        "total_tokens": 800
      }
    },
    "output_filtering": {
      "message": "Choice 0: Filtering module executed successfully on
content: no violations detected.",
      "data": {
        "choices": [
          {
            "index": 0,
            "azure_content_safety": {
              "Hate": 0,
              "SelfHarm": 0,
              "Sexual": 0,
              "Violence": 0
            }
          }
        ]
      }
    },
    "output_translation": {
      "message": "OTranslation module executed successfully on LLM output.",
      "data": {
        "choices": [
          {
            "index": 0,
            "message": {
              "role": "assistant",
              "content": "Die Schritte zur Erstkonfiguration in MASKED_ORG
umfassen Folgendes:\n\n1. Aktivieren Sie den Service in Cloud Foundry
mithilfe von Standardvorgehensweisen.\n2. Bereitstellung von MASKED_ORG aus
dem MASKED_ORG-Cockpit.\n3. Anlegen einer Serviceinstanz in der Umgebung
MASKED_PERSON.\n4. Binden Sie die Serviceinstanz an Ihre Anwendung, oder
legen Sie einen ServiceschlÃ¼ssel fÃ¼r die direkte Kommunikation mit der
Serviceinstanz an."
            },
            "finish_reason": "stop"
          }
```

```
        ]
      }
    }
  },
  "orchestration_result": {
    "id": "chatcmpl-BZu6RM8Z1wf74sgGu8gbtwxnoSETM",
    "object": "chat.completion",
    "created": 1747896839,
    "model": "gpt-4o-2024-08-06",
    "system_fingerprint": "fp_ee1d74bde0",
    "choices": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": "Die Schritte zur Erstkonfiguration in MASKED_ORG
umfassen Folgendes:\n\n1. Aktivieren Sie den Service in Cloud Foundry
mithilfe von Standardvorgehensweisen.\n2. Bereitstellung von MASKED_ORG aus
dem MASKED_ORG-Cockpit.\n3. Anlegen einer Serviceinstanz in der Umgebung
MASKED_PERSON.\n4. Binden Sie die Serviceinstanz an Ihre Anwendung, oder
legen Sie einen ServiceschlÃ¼ssel fÃ¼r die direkte Kommunikation mit der
Serviceinstanz an."
        },
        "finish_reason": "stop"
      }
    ],
    "usage": {
      "completion_tokens": 78,
      "prompt_tokens": 722,
      "total_tokens": 800
    }
  }
}
```


1.4.3.1.1 Grounding

The grounding module integrates external, domain-specific, or real-time data into generative AI workflows. By enriching pretrained models with contextually relevant information, grounding improves response accuracy and relevance beyond the models' general training data.

> **Note**
>
> Configuration of the grounding module is consistent across orchestration versions 1 and 2.

Pretrained generative models are trained on broad, general-purpose datasets. They do not have direct access to your organization's proprietary or real-time data.

Grounding addresses this limitation by:

- Connecting models to external data sources
- Retrieving relevant information at runtime
- Supplying that information as context for generation

This enables more accurate, business-specific, and up-to-date responses.

To index user-provided documents, the grounding module:

1. Splits the documents into smaller content chunks

2. Converts the chunks into vector representations (embeddings)
3. Stores the vectors in a vector database

For more information, see Pipelines API [page 438] and Vector API [page 477].

When a user submits a query:

1. The query is converted into a vector representation.
2. The vector database is searched for similar content.
3. Relevant information is returned and supplied to the generative model as context.

For more information, see Retrieval API [page 504].

Data Repositories

The following repositories and document types are supported:

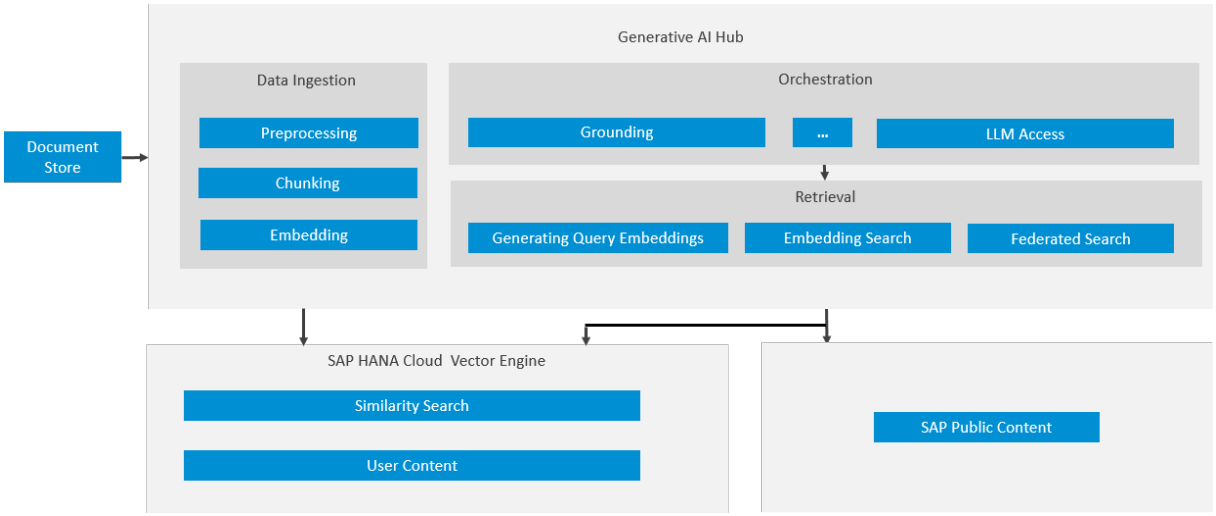
- Microsoft SharePoint
- AWS S3
- SFTP
- SAP Build Work Zone
- SAP Document Management service
- Service Now
- Google Drive

Document Repository	Document (File) Format	Content Refresh	Document Content	Maximum Number of Documents
Microsoft SharePoint	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT, JSON	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline
AWS S3	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT, JSON	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline
SFTP	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline
SAP Build Work Zone	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline
SAP Document Management service	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline

Document Repository	Document (File) Format	Content Refresh	Document Content	Maximum Number of Documents
Service Now	Knowledge Base Articles in HTML format	Daily	Plain Text	Maximum of 8000 Documents per Pipeline
Google Drive	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT,JSON	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline

Grounding Architecture

The grounding architecture in generative AI hub is as follows:



Prerequisites

To use the grounding module in the orchestration pipeline, you need to prepare the knowledge base in advance.

Generative AI hub offers multiple options for users to provide data (prepare a knowledge base):

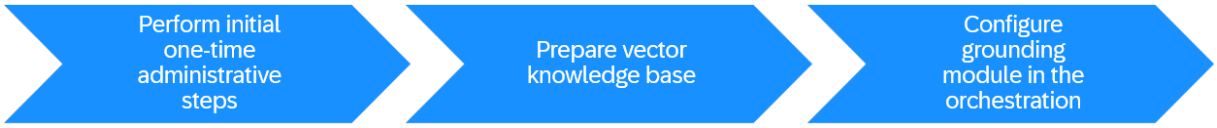
- For Option 1: Upload the documents to a supported data repository and run the data pipeline to vectorize the documents. For more information, see Pipelines API [page 438].
- For Option 2: Provide the chunks of document via Vector API directly. For more information, see Vector API [page 477].

To use the grounding module, choose from one of the following options.

Option 1: Upload the documents to the supported data repository and run Data Pipeline

In this case, the pipeline collects documents and segments the data into chunks. It generates embeddings, which are multidimensional representations of textual information, and stores them efficiently in a vector database.

The process for providing unstructured documents with the Pipeline API looks like this:



- Perform initial one-time administrative steps [page 395]
- Prepare vector knowledge base [page 395]
- Configure grounding module in the orchestration [page 395]

Hover over each action for a description. Click the action for more information.

Perform Initial One-Time Administrative Steps

Before you can prepare your data for the Pipeline API, you must create a resource group and a generic secret for grounding. For more information, see:

- Create a Resource Group for Grounding [page 397]
- Generic Secrets for Grounding [page 398]

Prepare Vector Knowledge Base

You configure the Pipeline API to read unstructured data from data repositories and store it in a vector database. Use the Pipeline API to:

1. Read unstructured documents from various data repositories. Segment the data into chunks and generate embeddings.
2. Store the multidimensional representations of the textual information in the vector database.
3. Provide a repository ID to access the data.

For more information, see Create a Document Grounding Pipeline Using the Pipelines API [page 440].

Configure Grounding Module in the Orchestration

In the orchestration pipeline, you add configuration for the grounding requests:

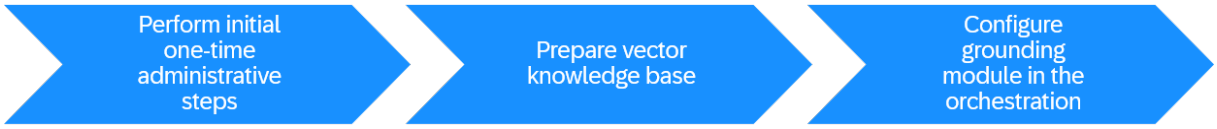
1. Create a grounding request configuration in the orchestration pipeline using the repository IDs and filters.
2. Run the orchestration pipeline and check that the response refers to the user data.

For more information, see Using the Grounding Module [page 510].

Option 2: Provide the chunks of document via Vector API directly

In this case, you provide chunks of data directly and store them using the Vector API.

The process for providing chunks of data with the Vector API looks like this:



- Perform initial one-time administrative steps [page 396]
- Prepare vector knowledge base [page 396]
- Configure grounding module in the orchestration [page 396]

Hover over each action for a description. Click the action for more information.

Perform Initial One-Time Administrative Steps

Before you can prepare your data for the Vector API, you must create a resource group. For more information, see Create a Resource Group for Grounding [page 397].

> **Note**
>
> When working with the Vector API, you don't need to create a generic secret for grounding.

Prepare Vector Knowledge Base

You provide chunks of information directly and store data in the vector database using the Vector API. Use the Vector API to:

1. Create collections.
2. Create documents by directly using the chunks of data provided by users.
3. Store data in the vector database.
4. Assign repository IDs to access the data.

For more information, see Preparing Data Using the Vector API [page 479].

Configure Grounding Module in the Orchestration

In the orchestration pipeline, you add configuration for the grounding requests:

1. Create a grounding request configuration in the orchestration pipeline using the repository IDs and filters.
2. Run the orchestration pipeline and check that the response refers to the user data.

For more information, see Using the Grounding Module [page 510].

Next Steps

Use the Retrieval API to search your data repositories.

For more information, see Retrieval API [page 504].

Related Information

Access the Grounding using the SAP AI SDK for Java
Access the Grounding using the SAP AI SDK for JavaScript

1.4.3.1.1.1 Prerequisites

1.4.3.1.1.1.1 Administrative Tasks

1.4.3.1.1.1.1.1 Create a Resource Group for Grounding

Prerequisites

Resource groups represent a virtual collection of related resources within the scope of one tenant. When a tenant is onboarded, the system immediately creates a default resource group. Tenant administrators can create or delete additional resource groups using the AI API. Tenants can map resource groups based on corresponding usage scenarios.

If a tenant uses resource groups to isolate scenario consumer tenants and those resource groups are deleted, the scenario consumers are deprovisioned. The service does not recognize the scenario consumer of the tenant. The standard XSUAA multitenancy model is followed.

For more information, see Scope of Resources.

Procedure

1. Create a resource group by sending a curl request, including the label that makes the resource group available to the grounding service.

   > **Note**
   >
   > Resource group Ids must be of length minimum: 3, maximum: 253. The first and last characters must be either a lowercase letter, an uppercase letter, or a number. Character entries from the second to penultimate can include a lower case letter, an upper case letter, a number, a period (.), or a hyphen (-). No other special characters are permitted.

   Set the following in the headers:

   - `AI_API_URL`: the base URL of your SAP AI Core environment.
   - `{{resource_group}}`: The AI resource group ID or document grounding management

- `{{access_token}}`: Your access token for SAP AI Core

```
curl --location --request POST "$AI_API_URL/v2/admin/resourceGroups"
--header "Authorization: Bearer {{access_token}}" \
--header 'Content-Type: application/json' \
--data-raw '{
  "resourceGroupId": "{{resource_group}}",
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    }
  ]
}'
```

Existing resource groups can be made available to the grounding service using a `PATCH` request, including the **`document-grounding`** label.

Next Steps

To provide data chunks for ingestion without registering a repository, use the Vector API data management API. For more information, see Vector API [page 477].

To register a repository to use as part of a grounding pipeline, create a generic secret for grounding for the repository of your choice. For more information, see Generic Secrets for Grounding [page 398].

Related Information

Edit a Resource Group [page 712]
Delete a Resource Group [page 714]

1.4.3.1.1.1.1.2 Generic Secrets for Grounding

The grounding module in SAP AI Core connects to external document repositories through generic secrets. These secrets store the credentials and connection details required to access repository content used in retrieval and grounding pipelines.

The grounding module in SAP AI Core supports the following data repositories:

- Microsoft SharePoint
- AWS S3
- SFTP
- SAP Build Work Zone
- SAP Document Management service
- Service Now
- Google Drive

Each repository type supports specific authentication methods. For Microsoft SharePoint, both `OAuth2Password` and `OAuth2ClientCredentials` authentication are supported. AWS S3 and SFTP use `NoAuthentication`. Although this authentication type is listed as `NoAuthentication`, the implementation still performs a basic authentication check for security purposes.

You must onboard at least one repository to enable grounding. For detailed onboarding instructions, see:

- Grounding Generic Secrets for Microsoft SharePoint [page 401]
- Grounding Generic Secrets for AWS S3 [page 405]
- Grounding Generic Secrets for SFTP [page 407]
- Grounding Generic Secrets for SAP Build Work Zone [page 410]
- Grounding Generic Secrets for SAP Document Management Service [page 412]
- Grounding Generic Secrets for ServiceNow [page 414]
- Create a Generic Secret for Google Drive [page 227]

The grounding module also supports metadata, which provides additional information about repositories for document indexing and contextualization. Metadata onboarding uses the `OAuth2ClientCredentials` authentication type and can be configured alongside any repository. For more information, see Generic Secrets for Contextualization with Metadata [page 419].

1.4.3.1.1.1.1.2.1 Create a Generic Secret for Google Drive

Creates a generic secret with base64-encoded credentials required to connect Google Drive to Document Grounding in SAP AI Core.

Overview

Your cloud storage credentials are managed using secrets. Secrets enable secure connections across directories and tools without exposing credentials.

Prerequisites

- A resource group is created for grounding. For more information, see Create a Resource Group for Grounding [page 397].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/admin/secrets`.
   Include the following headers in your request:
   - `AI_API_URL`: The base URL of your SAP AI Core environment.

- `{{access_token}}`: Your access token for SAP AI Core.

Set the following header:

- **`AI-Resource-Group`**: **`<resource-group-for-grounding>`**. The operation is performed at the resource-group level.

The following API request creates a generic secret with base64-encoded values, including client credentials and other required fields, to integrate grounding in the generative AI hub.

Request Fields

The following API request creates a generic secret with base64-encoded values required for integrating Google Drive with Document Grounding.

Field	Value
`name`	Name of the generic secret to be created
`authentication`	Base64-encoded value for **`OAuth2ClientCredentials`**
`description`	Base64-encoded description of the generic secret (optional)
`url`	Base64-encoded value of https://googleapis.com/drive/v3
`clientId`	Base64-encoded Client ID from the service account key JSON
`tokenServiceURL`	Base64-encoded value of https://oauth2.googleapis.com/token
`private_key`	Base64-encoded Private Key from the service account key JSON
`client_email`	Base64-encoded Client Email from the service account key JSON
`auth_uri`	Base64-encoded Authorization URI from the service account key JSON

> **Note**
>
> The following attributes are set by default and do not need to be passed in the API request:
>
> - `type`: HTTP
> - `proxyType`: Internet
> - `tokenServiceURLType`: Dedicated

Example

```
curl --request POST \
----url $AI_API_URL/v2/admin/secrets
--header 'AI-Resource-Group: <resource group created for grounding>' \
--header 'Authorization: Bearer <Bearer token>' \
--header 'Content-Type: application/json' \
--data-raw '{
```

```
  "name": "<GENERIC_SECRET_NAME>",
  "data": {
    "description": "<BASE64_ENCODED_DESCRIPTION>",
    "clientId": "<BASE64_ENCODED_CLIENT_ID>",
    "authentication": "T0F1dGgyQ2xpZW50Q3JlZGVudGlhbHM=",
    "url": "aHR0cHM6Ly9nb29nbGVhcGlzLmNvbS9kcml2ZS92Mw==",
    "tokenServiceURL": "aHR0cHM6Ly9vYXV0aDIuZ29vZ2xlYXBpcy5jb20vdG9rZW4=",
    "private_key": "<BASE64_ENCODED_PRIVATE_KEY>",
    "client_email": "<BASE64_ENCODED_CLIENT_EMAIL>",
    "auth_uri": "<BASE64_ENCODED_AUTH_URI>"
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
      "value": "GoogleDrive"
    }
  ]
}'
```

Next Steps

After creating the generic secret, create a Create a Document Grounding Pipeline Using the Pipelines API [page 440].

1.4.3.1.1.1.1.2.2 Grounding Generic Secrets for Microsoft SharePoint

Your cloud storage credentials are managed using secrets. Secrets are a means of allowing and controlling connections across directories and tools, without compromising your credentials.

Prerequisites

- You've prepared your SharePoint integration. For more information, see Prepare SharePoint Integration with Joule.
- You've created a resource group for grounding. For more information, see Create a Resource Group for Grounding [page 397].

Procedure

1. Send a POST request and enter the URL `{{apiurl}}/v2/admin/secrets`.
   Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

   The following API request is used to create a generic secret with base64-encoded values, such as client credentials and other necessary fields, for integrating grounding in the generative AI hub. It can use OAuth2Password or OAuth2ClientCredentials to generate access tokens.

> **Note**
>
> In *API permissions*:
> - If you configure access from SAP BTP with **`OAuth2Password`** authentication, use the permission name **`Sites.Read.All`** with type **`Delegated`**
> - If you configure access from SAP BTP with **`OAuth2ClientCredentials`** authentication, use the permission name **`Sites.Selected`** with type **`Application`**
>
> For more information, see Integrating Joule with SAP Solutions: Configure Access from SAP BTP.

Examples

Example with OAuth2Password:

The `name` attribute is written without hyphens to make it simple to consume as a Unix environment variable later. It's written in capitals as is convention.

Field	Value
`name`	Name of the generic secret to be created
`description`	Base64 encoded value for the description of the generic secret to be created
`url`	Base64 encoded value for **`https://graph.microsoft.com`**
`authentication`	Base64 encoded value for **`OAuth2Password`**

Field	Value
`user`	Base64 encoded value for your Microsoft SharePoint technical user
`password`	Base64 encoded value for your Microsoft SharePoint password
`clientId`	Base64 encoded value for your Microsoft SharePoint Application credentials
`clientSecret`	Base64 encoded value for your Microsoft SharePoint Application credentials
`tokenServiceURL`	Base64 encoded value for `https://login.microsoftonline.com/<Microsoft EntraIDtenantidentifier>/oauth2/v2.0/token`

> **Note**
>
> The following attributes are set as default, and don't need to be passed during the API call:
>
> - `type`: `"HTTP"`
> - `proxyType`: `"Internet"`
> - `Scope`: `https://graph.microsoft.com/.default`

```
curl --location --request POST "$AI_API_URL/v2/admin/secrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <resource group created for grounding>' \
--data-raw '{
  "name": "<generic secret name>",
  "data": {
    "description": "<description of generic secret>",
    "clientId": "<client id>",
    "authentication": "T0F1dGgyUGFzc3dvcmQ=",
    "tokenServiceURL": "<token service url>",
    "password": "<password>",
    "url": "aHR0cHM6Ly9ncmFwaC5taWNyb3NvZnQuY29t",
    "user": "<user>",
    "clientSecret": "<client secret>",
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
     },
     {
      "key": "ext.ai.sap.com/documentRepositoryType",
       "value": "MSSharePoint"
    }
  ]
}'
```

Example with OAuth2ClientCredentials:

The `name` attribute is written without hyphens to make it simple to consume as a Unix environment variable later. It's written in capitals as is convention.

Field	Value
`name`	Name of the generic secret to be created
`description`	Base64 encoded value for the description of the generic secret to be created
`url`	Base64 encoded value for **`https://sap.sharepoint.com/sites/<site-name>`**
`authentication`	Base64 encoded value for **`OAuth2ClientCredentials`**
`clientId`	Base64 encoded value for your Microsoft SharePoint Application credentials
`clientSecret`	Base64 encoded value for your Microsoft SharePoint Application credentials
`tokenServiceURL`	Base64 encoded value for `https://login.microsoftonline.com/` **`<Microsoft EntraIDtenantidentifier>`**`/oauth2/v2.0/token`

> **Note**
>
> The following attributes are set as default, and don't need to be passed during the API call:
>
> - `type`: `"HTTP"`
> - `proxyType`: `"Internet"`
> - `tokenServiceURLType`: `"Dedicated"`
> - `Scope`: `https://graph.microsoft.com/.default`

> **Sample Code**

```
curl --request POST \
--url $AI_API_URL/v2/admin/secrets \
--header 'AI-Resource-Group: <resource group created for grounding>' \
--data-raw '{
  "name": "<generic secret name>",
  "data": {
    "description": "<description of generic secret>",
    "clientId": "<client id>",
    "authentication": "T0F1dGgyQ2xpZW50Q3JlZGVudGlhbHM=",
    "tokenServiceURL": "<token service
url>",
    "url":
"aHR0cHM6Ly9zYXAuc2hhcmVwb2ludC5jb20vc2l0ZXMvPHNpdGUtbmFtZT4=",

    "clientSecret": "<client secret>",
  },
   "labels": [
    {
```

```
      "key": "ext.ai.sap.com/document-grounding",
        "value": "true"
      },
      {
       "key": "ext.ai.sap.com/documentRepositoryType",
       "value": "MSSharePoint"
     }
   ]
}'
```

Next Steps

After registering your repository with grounding, you can use the document grounding APIs to prepare and store your vectors. For more information, see Pipelines API [page 438].

Related Information

Update a Generic Secret [page 715]
Delete a Generic Secret [page 718]

1.4.3.1.1.1.1.2.3 Grounding Generic Secrets for AWS S3

Your cloud storage credentials are managed using secrets. Secrets are a means of allowing and controlling connections across directories and tools, without compromising your credentials.

Context

You have created a resource group for grounding. For more information, see Create a Resource Group for Grounding [page 397].

Procedure

1. Send a POST request and enter the URL `{{apiurl}}/v2/admin/secrets`.
   Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN

Header	Value
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

The following API request is used to create a generic secret with base64-encoded values, such as client credentials and other necessary fields, for integrating grounding in the generative AI hub.

Example

The `name` attribute is written without hyphens to make it simple to consume as a Unix environment variable later. It is written in capitals as is convention.

Field	Value
`name`	Name of the generic secret to be created
`url`	Base64-encoded value in the format **`https://s3.<region>.amazonaws.com`**
`authentication`	Base64-encoded value for **`NoAuthentication`**
`description`	Base64-encoded value for the description of the generic secret to be created
`access_key_id`	Base64-encoded value for access key id
`bucket`	Base64-encoded value for AWS S3 bucket name
`host`	Base64-encoded value for AWS S3 host
`region`	Base64-encoded value for AWS S3 region
`secret_access_key`	Base64-encoded value for AWS S3 credentials
`username`	Base64-encoded value for AWS S3 credentials

> **Note**
>
> The following attributes are set as default and do not need to be passed during the API call:
>
> - `type`: `"HTTP"`
> - `proxyType`: `"Internet"`

```
curl --location --request POST "$AI_API_URL/v2/admin/secrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <resource group created for grounding>' \
--data-raw '{
```

```
  "name": "<generic secret name>",
  "data": {
    "url": "<url>",
    "authentication": "Tm9BdXRoZW50aWNhdGlvbg==",
    "description": "<description of generic secret>",
    "access_key_id": "<access key id>",
    "bucket": "<bucket>",
    "host": "<host>",
    "region": "<region>",
    "secret_access_key": "<secret access key>",
    "username": "<username>"
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
      "value": "S3"
    }
  ]
}'
```

Next Steps

After registering your repository with grounding, you can use the document grounding APIs to prepare and store your vectors. For more information, see Pipelines API [page 438].

Related Information

Update a Generic Secret [page 715]
Delete a Generic Secret [page 718]
Making requests using the REST API

1.4.3.1.1.1.1.2.4 Grounding Generic Secrets for SFTP

Your cloud storage credentials are managed using secrets. Secrets are a means of allowing and controlling connections across directories and tools, without compromising your credentials.

Prerequisites

You have created a resource group for grounding. For more information, see Create a Resource Group for Grounding [page 397].

Procedure

1. Send a POST request and enter the URL `{{apiurl}}/v2/admin/secrets`.
   Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

   The following API request is used to create a generic secret with base64-encoded values, such as client credentials and other necessary fields, for integrating grounding in the generative AI hub.

Example

The `name` attribute is written without hyphens to make it simple to consume as a Unix environment variable later. It is written in capitals as is convention.

Field	Value
`name`	Name of the generic secret to be created
`url`	Base64-encoded SFTP endpoint URL in the format **`https://<hostname>:<port>`**, where `<hostname>` and `<port>` represent the SFTP server host and port. Use this format when configuring the endpoint. For example, `https://sftp.example.com:22` **Note** Although the connection uses the SFTP protocol internally, the endpoint must be specified using the `https://` format **Restriction** Do not use `sftp://<hostname>:<port>`
`authentication`	Base64 encoded value for **`NoAuthentication`**
`description`	Base64 encoded value for the description of the generic secret to be created
`user`	Base64 encoded value for SFTP credentials

Field	Value
`password`	Base64 encoded value for SFTP credentials
`public_key`	The Base64-encoded SFTP server host public key is used to verify the server identity during connection setup and must be provided in addition to the SFTP user credentials.
`public_key_type`	Base64 encoded value for SFTP public key type: `ssh-ed25519` and `ssh-rsa`.

> **Note**
>
> The following attributes are set as default, and do not need to be passed during the API call:
>
> - `type`: `"HTTP"`
> - `proxyType`: `"Internet"`

Retrieve the SFTP Server Public Key

You can retrieve the SFTP server host public key using standard SSH utilities such as `ssh-keyscan`.

For more information, see https://man.openbsd.org/ssh-keyscan .

```
curl --location --request POST "$AI_API_URL/v2/admin/secrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <resource group created for grounding>' \
--data-raw '{
  "name": "<generic secret name>",
  "data": {
    "url": "<url>",
    "authentication": "Tm9BdXRoZW50aWNhdGlvbg==",
    "description": "<description of generic secret>",
    "password": "<password>",
    "user": "<user>",
    "public_key": "<public key>",
    "public_key_type": "<public key type>"
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
      "value": "SFTP"
    }
    }
  ]
}'
```

Next Steps

After registering your repository with grounding, you can use the document grounding APIs to prepare and store your vectors. For more information, see Pipelines API [page 438].

Related Information

Update a Generic Secret [page 715]
Delete a Generic Secret [page 718]

1.4.3.1.1.1.1.2.5 Grounding Generic Secrets for SAP Build Work Zone

Your cloud storage credentials are managed using secrets. Secrets are a means of allowing and controlling connections across directories and tools, without compromising your credentials.

Prerequisites

You've created a resource group for grounding. For more information, see Create a Resource Group for Grounding [page 397].

Procedure

1. Send a POST request and enter the URL `{{apiurl}}/v2/admin/secrets`.
   Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

   The following API request is used to create a generic secret with base64-encoded values, such as client credentials and other necessary fields, for integrating grounding in the generative AI hub. It can use OAuth2ClientCredentials with client secret.

Example

The `name` attribute is written without hyphens to make it simple to consume as a Unix environment variable later. It's written in capitals as is convention.

Field	Value
`name`	Name of the generic secret to be created
`description`	Base64 encoded value for the description of the generic secret to be created
`url`	Base64 encoded value for your SAP Build Work Zone URL
`authentication`	Base64 encoded value for **`OAuth2ClientCredentials`**
`clientId`	Base64 encoded value for client ID
`clientSecret`	Base64 encoded value for client secret
`tokenServiceURL`	Base64 encoded value for token service URL for client secret based authentication

> **Note**
>
> The following attributes are set as default, and don't need to be passed during the API call:
>
> - `type`: `"HTTP"`
> - `proxyType`: `"Internet"`
> - `tokenServiceURLType`: `"Dedicated"`

```
curl --request POST \
  --url $AI_API_URL/v2/admin/secrets \
  --header 'ai-resource-group: {{resource_group}}' \
  --header 'content-type: application/json' \
  --data '{
    "name": "<name-of-secret>",
    "data": {
    "description": "<any string base64 encoded>",
    "authentication": "T0F1dGgyQ2xpZW50Q3JlZGVudGlhbHM=",
    "url": "< base64 encoded SAP Build Work Zone-server-url>",
    "clientId": "< base64 encoded client-id>",
    "clientSecret": "< base64 encoded client-secret>",
    "tokenServiceURL": "< base64 encoded token-service-url>"
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
      "value": "WorkZone"
    }
  ]
}'
```

Next Steps

After registering your repository with grounding, you can use the document grounding APIs to prepare and store your vectors. For more information, see Pipelines API [page 438].

Related Information

Update a Generic Secret [page 715]
Delete a Generic Secret [page 718]
Integration With Document Grounding SAP Build Work Zone, Advanced Edition

1.4.3.1.1.1.1.2.6 Grounding Generic Secrets for SAP Document Management Service

Prerequisites

You have created a resource group for grounding. For more information, see Create a Resource Group for Grounding [page 397].

Procedure

1. Send a POST request and enter the URL `{{apiurl}}/v2/admin/secrets`.
   Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

   The following API request is used to create a generic secret with base64-encoded values, such as client credentials and other necessary fields, for integrating grounding in the generative AI hub. It can use OAuth2ClientCredentials with client secret.

Example

The `name` attribute is written without hyphens to make it simple to consume as a Unix environment variable later. It is written in capitals as is convention.

Field	Value
`name`	Name of the generic secret to be created
`description`	Base64 encoded value for the description of the generic secret to be created
`url`	Base64 encoded value for your AI Bridge Service URL and repository path. For example, `https://sdm-ai-bridge.cfapps.<region>.hana.ondemand.com/rest/v1/repositories/<repoID>/metadata`.
`authentication`	Base64 encoded value for **`OAuth2ClientCredentials`**
`clientId`	Base64 encoded value for client ID
`clientSecret`	Base64 encoded value for client secret
`tokenServiceURL`	Base64 encoded value for token service URL for client secret based authentication

> **Note**
>
> The following attributes are set as default, and do not need to be passed during the API call:
>
> - `type`: `"HTTP"`
> - `proxyType`: `"Internet"`
> - `tokenServiceURLType`: `"Dedicated"`

```
curl --request POST \
  --url $AI_API_URL/v2/admin/secrets \
  --header 'ai-resource-group: {{resource_group}}' \
  --header 'content-type: application/json' \
  --data '{
    "name": "<name-of-secret>",
    "data": {
    "description": "<any string base64 encoded>",
    "authentication": "T0F1dGgyQ2xpZW50Q3JlZGVudGlhbHM=",
    "url": "<base64 encoded AI Bridge Service URL>",
    "clientId": "< base64 encoded client-id>",
    "clientSecret": "< base64 encoded client-secret>",
    "tokenServiceURL": "< base64 encoded token-service-url>"
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
```

```
      "value": "SDM"
    }
  ]
}'
```

Next Steps

After registering your repository with grounding, you can use the document grounding APIs to prepare and store your vectors. For more information, see Pipelines API [page 438].

Related Information

Update a Generic Secret [page 715]
Delete a Generic Secret [page 718]
SAP Document Management Service
Access the Document Grounding using the SAP AI SDK for Java
Access the Document Grounding using the SAP AI SDK for JavaScript

1.4.3.1.1.1.1.2.7 Grounding Generic Secrets for ServiceNow

Your cloud storage credentials are managed using secrets. Secrets are a means of allowing and controlling connections across directories and tools, without compromising your credentials.

Prerequisites

You have created a resource group for grounding. For more information, see Create a Resource Group for Grounding [page 397].

Procedure

1. Send a POST request and enter the URL `{{apiurl}}/v2/admin/secrets`.
   Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN

Header	Value
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

The following API request is used to create a generic secret with base64-encoded values, such as client credentials and other necessary fields, for integrating grounding in the generative AI hub. It can use OAuth2ClientCredentials with client secret.

Example

The `name` attribute is written without hyphens to make it simple to consume as a Unix environment variable later. It is written in capitals as is convention.

Field	Value
`name`	Name of the generic secret to be created
`description`	Base64 encoded value for the description of the generic secret to be created
`url`	Base64 encoded value for your ServiceNow instance URL
`authentication`	Base64 encoded value for **`ClientCertificateAuthentication`**
`certificateType`	Base64 encoded value for pfx
`certificateContent`	Base64 encoded value for your certificate content
`certificatePassword`	Base64 encoded value for your certificate password

> **Note**
>
> The following attributes are set as default, and do not need to be passed during the API call:
>
> - `type`: `"HTTP"`
> - `proxyType`: `"Internet"`

```
 curl --location --request POST $AI_API_URL/v2/admin/secrets \
--header "Authorization: Bearer $TOKEN" \
--header "Content-Type: application/json" \
--header "AI-Resource-Group: <resource-group-for-grounding>" \
--data-raw ''{
  "name": "<GENERIC_SECRET_NAME>",
  "data": {
    "authentication": "Q2xpZW50Q2VydGlmaWNhdGVBdXRoZW50aWNhdGlvbg==",
    "description": "<BASE64_ENCODED_DESCRIPTION>",
    "certificateType": "cGZ4",
    "certificateContent": "<BASE64_ENCODED_CERT_CONTENT>",
```

```
    "certificatePassword": "<BASE64_ENCODED_CERT_PASSWORD>",
    "url": "<BASE64_ENCODED_SERVICENOW_URL>"
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
      "value": "ServiceNow"
    }
  ]
}'
```

Next Steps

After registering your repository with grounding, you can use the document grounding APIs to prepare and store your vectors. For more information, see Pipelines API [page 438].

Related Information

Update a Generic Secret [page 715]
Delete a Generic Secret [page 718]

1.4.3.1.1.1.1.3 Contextualization Using Metadata

1.4.3.1.1.1.1.3.1 Prepare your Metadata API Server

The metadata server enables grounding to process documents from your repositories, enriched with metadata for improved contextualization. To incorporate metadata into grounding, you must prepare, deploy and host a metadata server. The **default** path is `/v1/dg-pipeline/metadata` and the HTTP method is GET.

For metadata servers hosted at the default path, you only need to specify the **base** URL, for example, https://metadata-server.com in the **URL** field of the generic secret.

For metadata servers not hosted at the default path, you need to specify the **full** URL, for example, https://metadata-server.com/custom/path/to/metadata in the **URL** field of the generic secret.

Metadata API Response Structure

Your metadata server must return a response in the following format:

Output Code

```
{
  "documentsCount": 100,
  "value": [
    {
      "metadata": [
        {
          "key": "domainId",
          "value": ["PUBLIC", "PRIVATE"],
          "matchMode": "ANY"
        },
        {
          "key": "department",
          "value": ["HR", "Finance", "Engineering"],
          "matchMode": "ALL"
        }
      ],
      "documents": [
        {
          "id": "3893fe5e-9b22-4eb3-ad12-6a8b4b38f25h",
          "viewLocation": "/path/to/view/doc",
          "metadata": [
            {
              "key": "domainId",
              "value": ["PRIVATE"]
            },
            {
              "key": "department",
              "value": ["HR", "Finance"],
              "matchMode": "ALL"
            }
          ],
          "downloadLocation": "/path/to/download/doc",
          "lastUpdatedTimestamp": "2024-02-25T12:00:00Z"
        }
        // ... more documents
      ]
    }
    // ... more document groups
  ]
}
```

The top-level value is an array of document groups.

Each document group contains group-level `metadata` and a `documents` array.

Each document must have the following:

- A unique `id`
- `viewLocation`: the file location in the repository
- `lastUpdatedTimestamp`: the timestamp of the last update

Optionally, documents can also have the following;

- `metadata`: your choice of labels
- `downloadLocation`: the location from that the file can be downloaded from

The metadata keys and values are your own meaningful custom strings.

`matchMode` must be `ALL` to match all metadata values, or `ANY` to match any single value.

Authentication Requirements

Your metadata API must be secured using OAuth2ClientCredentials flow. You can use one of the following:

- Client secret-based authentication
- Mutual TLS (mTLS) with client certificates

For more information, see Generic Secrets for Contextualization with Metadata [page 419].

Ensure your server validates the incoming access token before serving the metadata.

Pagination

Pagination is optional.

If pagination is included, the following query parameters must be supported:

Parameter	Type	Description
`$top`	Integer	Max number of documents to return.
`$skip`	Integer	Number of documents to skip.
`$count`	Boolean	`true`: returns the total number of documents available on the server in the `documentsCount` field. `false`: returns the number of documents included in the current response in the `documentsCount` field.

Pagination requirements

Pagination must be applied across all document groups, meaning the API returns `$top`, `$count` and skip the first `$skip` documents globally, regardless of which group they belong to.

Document groups with no documents must be removed from the response.

The API must not return more documents than specified by `$top`.

The generic secret for the metadata server can include a `pageSize` attribute to set `$top`. For more information, see Generic Secrets for Contextualization with Metadata [page 419]

Example

A GET request to the endpoint `GET <BASE_URL>/v1/dg-pipeline/metadata?$top=2&$skip=1&$count=true` should return the following response:

Output Code

```
{
  "documentsCount": 5, // Total documents available on server (since
$count=true)
  "value": [
    {
      "metadata": [
```

```
        { "key": "domainId", "value": ["PUBLIC", "PRIVATE"], "matchMode":
"ANY" },
        { "key": "department", "value": ["HR", "Finance", "Engineering"],
"matchMode": "ALL" }
      ],
      "documents": [
        {
          "id": "doc-002-id",
          "viewLocation": "/path/to/view/doc2",
          "metadata": [
            { "key": "domainId", "value": ["PUBLIC"], "matchMode": "ANY" },
            { "key": "department", "value": ["HR", "Finance"], "matchMode":
"ALL" }
          ],
          "downloadLocation": "/path/to/download/doc2",
          "lastUpdatedTimestamp": "2024-02-26T12:00:00Z"
        },
        {
          "id": "doc-003-id",
          "viewLocation": "/path/to/view/doc3",
          "metadata": [
            { "key": "domainId", "value": ["PRIVATE"]}
          ],
          "downloadLocation": "/path/to/download/doc3",
          "lastUpdatedTimestamp": "2024-02-27T12:00:00Z"
        }
      ]
    }
  ]
}
```

Deployment

Now that you've prepared your application, you can deploy it. For more information, see SAP BTP Deploy an Application, DAP BTP Deploying to the Cloud Foundry Environment and Create an Application with Cloud Foundry Python Buildpack BTP Tutorial.

1.4.3.1.1.1.1.3.2 Generic Secrets for Contextualization with Metadata

Prerequisites

You have created a resource group for grounding. For more information, see Create a Resource Group for Grounding [page 397].

Procedure

1. Send a POST request and enter the URL `{{apiurl}}/v2/admin/secrets`.
   Include the following headers in your request:
   - `AI_API_URL`: the base URL of your SAP AI Core environment.
   - `{{access_token}}`: Your access token for SAP AI Core

   Choose one from the following headers to set your scope:
   - **`AI-Tenant-Scope`** : **`true`**. The operation is performed at the main tenant level.
   - **`AI-Resource-Group`** : **`<resource-group-for-grounding>`**. The operation is performed at the resource-group level.

   The following API request is used to create a generic secret with base64-encoded values, such as client credentials and other necessary fields, for integrating grounding in the generative AI hub. It can use OAuth2ClientCredentials with client secret or client certificate.

The `name` attribute is written without hyphens to make it simple to consume as a Unix environment variable later. It is written in capitals as is convention.

Common Fields

Field	Value
`name`	Name of the generic secret to be created
`description`	Base64 encoded value for the description of the generic secret to be created
`url`	Base64 encoded value for the metadata API server URL
`authentication`	Base64 encoded value for **`OAuth2ClientCredentials`**
`clientId`	Base64 encoded value for client ID
`pageSize` (optional)	If pagination is required, enter a Base64 encoded value of integer from 1 to 1000. **⚠ Restriction** Pagination and the page size attribute are only supported if you have a metadata server with pagination implemented. For more information, see Prepare your Metadata API Server [page 416].

With Client Secret

Field	Value
`clientSecret`	Base64 encoded value for client secret
`tokenServiceURL`	Base64 encoded value for token service URL for client secret based authentication

With Client Certificate

Field	Value
`certificateType`	Base64 encoded value for `P12` or `PEM`
`certificateContent`	Base64 encoded value for certificate content
`certificatePassword` (optional)	Base64 encoded value for password of the certificate if set
`tokenServiceURL`	Base64 encoded value for token service URL for certificate based authentication
`tokenServiceBodyParameters` (optional)	Base64 encoded value for valid JSON object with key-value pair

> **Note**
>
> The following attributes are set as default, and do not need to be passed during the API call:
>
> - `type`: `"HTTP"`
> - `proxyType`: `"Internet"`
> - `tokenServiceURLType`: `"Dedicated"`

Examples

Example with Client Secret:

> **Sample Code**

```
curl --location --request POST "$AI_API_URL/v2/admin/secrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <resource group created for grounding>' \
--data-raw '{
  "name": "<generic secret name>",
  "data": {
    "url": "<url>",
    "description": "<description of generic secret>",
    "authentication": "T0F1dGgyQ2xpZW50Q3JlZGVudGlhbHM=",
    "clientId": "<client id>",
    "clientSecret": "<client secret>",
    "tokenServiceURL": "<token service url for client secret based
authentication>",
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
      "value": "MetaData"
    }
  ]
```

```
}'
```

Example with Client Secret and Pagination:

Sample Code

```
curl --location --request POST "$AI_API_URL/v2/admin/secrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <resource group created for grounding>' \
--data-raw '{
  "name": "<generic secret name>",
  "data": {
    "url": "<url>",
    "description": "<description of generic secret>",
    "authentication": "T0F1dGgyQ2xpZW50Q3JlZGVudGlhbHM=",
    "clientId": "<client id>",
    "clientSecret": "<client secret>",
    "tokenServiceURL": "<token service url for client secret based
authentication>",
    "pageSize": "<Base64 encoded integer 1-1000>"
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
      "value": "MetaData"
    }
  ]
}'
```

Example with Client Certificate:

Sample Code

```
curl --location --request POST "$AI_API_URL/v2/admin/secrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <resource group created for grounding>' \
--data-raw '{
  "name": "<generic secret name>",
  "data": {
    "url": "<url>",
    "description": "<description of generic secret>",
    "authentication": "T0F1dGgyQ2xpZW50Q3JlZGVudGlhbHM=",
    "clientId": "<client id>",
    "certificateType": "UDEyCg== [P12] / UEVNCg== [PEM]",
    "certificateContent": "<certificate content>",
    "certificatePassword": "<certificate password>",
    "tokenServiceBodyParameters": "<valid json with key-value pairs>",
    "tokenServiceURL": "<token service url for certificate based
authentication>
```

```
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
      "value": "MetaData"
    }
  ]
}'
```

Example with Client Certificate and Pagination:

Sample Code

```
curl --location --request POST "$AI_API_URL/v2/admin/secrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <resource group created for grounding>' \
--data-raw '{
  "name": "<generic secret name>",
  "data": {
    "url": "<url>",
    "description": "<description of generic secret>",
    "authentication": "T0F1dGgyQ2xpZW50Q3JlZGVudGlhbHM=",
    "clientId": "<client id>",
    "certificateType": "UDEyCg== [P12] / UEVNCg== [PEM]",
    "certificateContent": "<certificate content>",
    "certificatePassword": "<certificate password>",
    "tokenServiceBodyParameters": "<valid json with key-value pairs>",
    "tokenServiceURL": "<token service url for certificate based
authentication>"
    "pageSize": "<integer 1-1000>"
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
      "value": "MetaData"
    }
  ]
}'
```

1.4.3.1.1.1.2 Data Management APIs

Using the Document Grounding APIs, you can automate the process of retrieving, preprocessing, and embedding documents from supported data sources into the SAP HANA Vector Store. These APIs handle the end-to-end flow of documents, including document fetching, text chunking, and embedding via the Vector API, without requiring you to manage each step manually.

For document processing and vector generation, the following options are available:

- For use cases where documents are stored in a repository, the **Pipelines API** is recommended.
  The **Pipelines API** creates data management pipelines that fetch documents from a supported data source, preprocess and chunk those documents, and store their semantic embeddings in the HANA Vector Store. For more information, see Pipelines API [page 438].
- For use cases where documents are uploaded and managed directly, the **Vector API** is recommended.
  The **Vector API** preprocesses documents into chunks and stores their semantic embeddings in collections. For more information, see Vector API [page 477].

The **Retrieval API** performs similarity searches on the vector database for information retrieval. It can be used for repositories of documents that have been processed using the Pipelines API, or for collections that have been processed using the Vector API. For more information, see Preparing Data Using the Vector API [page 479].

> → Tip
>
> If you use the Pipelines API, you don't also need to call the Vector API. After the data is embedded, you can directly use the Retrieval API to query the vector store for relevant results.

For more information about the API, such as request formats, response schemas, authentication headers, and supported operations, see Grounding APIs in the Business Accelerator Hub.

Rate Limits

Tenant Level

The following rate limits apply:

Operation Type	Call Type	Rate Limit
write_pipeline	POST – Create pipeline	5 requests per minute
write	PUT, POST, DELETE, PATCH	6 requests per second
read	GET	1200 requests per minute

Resource Group Level

The following rate limits apply:

Pipelines

Operation Type	Call Type	Rate Limit
write_pipeline	POST – Create pipeline	5 requests per minute
write	PUT, POST, DELETE, PATCH	6 requests per second
read	GET	1200 requests per minute

Metadata Management

Operation Type	Call Type	Rate Limit
write_pipeline	POST – Create pipeline	5 requests per minute
write	PUT, POST, DELETE, PATCH	6 requests per second
read	GET	1200 requests per minute

Vector

Endpoint	Call Type	Rate Limit
`$AI_API_URL/v2/lm/document-grounding/vector/search`	POST	250 requests per minute
`$AI_API_URL/v2/lm/document-grounding/vector/documents`	DELETE	
`$AI_API_URL/v2/lm/document-grounding/vector/documents/metadata`	PATCH	
`$AI_API_URL/v2/lm/document-grounding/vector/chunks/metadata`	PATCH	
`$AI_API_URL/v2/lm/document-grounding/vector/collections`	GET	
`$AI_API_URL/v2/lm/document-grounding/vector/collections`	POST	
`$AI_API_URL/v2/lm/document-grounding/vector/collections/metadata`	PATCH	
`$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}`	DELETE	
`$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents`	GET	
`$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents`	POST	

Endpoint	Call Type	Rate Limit
`$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents`	PATCH	
`$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents/{{documentId}}`	GET	
`$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents/{{documentId}}`	DELETE	
`$AI_API_URL/v2/lm/document-grounding/vector/collections/{{id}}/creationStatus`	GET	500 requests per minute
`$AI_API_URL/v2/lm/document-grounding/vector/collections/{{id}}/deletionStatus`	GET	
`$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}`	GET	

Retrieval

Endpoint	Call Type	Rate Limit
`$AI_API_URL/v2/lm/document-grounding/retrieval/dataRepositories`	GET	250 requests per minute
`$AI_API_URL/v2/lm/document-grounding/retrieval/search`	POST	
`$AI_API_URL/v2/lm/document-grounding/retrieval/dataRepositories/{{repositoryId}}`	GET	500 requests per minute

Capacity Quotas and Input Limits

Vector Ingestion – Capacity Limits

The following capacity limits apply:

Category	Description	Limit Type	Limit
Collections	Max collections	Quota	55,000
Documents	Max documents per collection	Quota	100,000
Documents	Max documents per ingestion request	Input Limit	100
Chunks	Max chunks per document	Quota	200,000

Metadata Limits (Ingestion and Query Time)

The following metadata limits apply:

Metadata Type	Limit Type	Limit
Collection Metadata	Quota / Input Limit	500 / 5,000
Document Metadata	Quota / Input Limit	500 / 5,000
Chunk Metadata	Quota / Input Limit	500 / 5,000

Vector Search – Query Limits

The following query limits apply:

Category	Description	Limit Type	Limit
Filters	Max filters per query	Input Limit	15
Collections	Max collection IDs per query	Input Limit	5,000
Results	Max chunk results per query	Input Limit	100
Results	Max document results per query	Input Limit	10

Retrieval Search – Query Limits

The following query limits apply:

Category	Description	Limit Type	Limit
Filters	Max filters per query	Input Limit	15
Collections	Max collection IDs per query	Input Limit	5,000
Results	Max chunk results per query	Input Limit	100
Results	Max document results per query	Input Limit	10

1.4.3.1.1.1.2.1 Metadata Management API

Overview

The Metadata Management API provides endpoints for managing metadata configurations and document metadata within the document grounding pipelines. You can create and manage metadata configurations for various data repositories, enumerate documents, and update document metadata in batch operations.

Key Capabilities

The Metadata Management API enables you to:

- Create and manage metadata configurations for different repository types.
- Retrieve metadata configurations with pagination support and detailed status information.
- Enumerate documents automatically from configured repositories.
- List and filter documents within metadata configurations using path-based queries.
- Update document metadata in batch for up to 1,000 documents per request.
- Manage the lifecycle of your metadata configurations.

API Request Format

When you build your API requests, follow the format outlined in the *Pipelines* section of the Grounding API. For more information, see Grounding API.

The endpoint for the pipelines API is `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations`.

1.4.3.1.1.1.2.1.1 Create a Metadata Configuration

Creates a new metadata configuration and triggers document enumeration for the configured repository.

Context

You can create a metadata configuration for reuse across your pipelines, instead of defining a metadata configuration for each pipeline.

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- You have created a generic secret. For more information, see Generic Secrets for Grounding [page 398].
- You've added your grounding documents to your document store and you're using a supported data source and document type. For more information, see Pipelines API [page 438].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations`.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Populate the sample code with the following values:

Field	Value
name	A unique identifier for your configuration
destinationName	The name of the generic secret for your data repository
dataRepositoryType	Your data repository type. Choose from: • `MSSharePoint` • `S3` • `SFTP`
includePaths	The paths to the documents to be included, in a comma separated list.
labels	Your metadata labels in key-value pairs.

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/metadata/
configurations \
  --header 'AI-Resource-Group: <resource_group>' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer $TOKEN' \
  --data '{
      "name": "config-name",
      "destinationName": "destination-name",
      "dataRepositoryType": "<data repository type>",
```

```
      "includePaths": [
        "/path/one",
        "/path/two"
      ],
      "labels": [
        {
          "key": "department",
          "value": "finance"
        }
      ]
    }'
```

Results

Successful responses return code **202** and include a success message.

The response includes details of your metadata configuration resource, including the **metadata configuration ID**.

Document enumeration is performed asynchronously after the request is accepted.

Next Steps

You can reference your metadata configuation as part of a pipeline by referencing the metadata configuration ID when you create your pipeline. For more information, see Create a Pipeline using a Metadata Configuration [page 462].

1.4.3.1.1.1.2.1.2 List Metadata Configurations

Retrieves a paginated list of all available metadata configurations.

Procedure

1. Send a GET request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations`.
   You can include optional parameters for pagination support. The following query parameters are supported:

Parameter	Type	Description
`$top`	Integer	Max number of documents to return.

Parameter	Type	Description
`$skip`	Integer	Number of documents to skip.
`$count`	Boolean	`true`: returns the total number of documents available on the server in the `documentsCount` field. `false`: returns the number of documents included in the current response in the `documentsCount` field.

For example: `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations?top=10&skip=0`

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/metadata/
configurations \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

Results

Successful responses return code **200** and include details of your **metadata configurations** in JSON format.

Output Code

```
{
  "count": 1,
  "resources": [
    {
      "id": "uuid",
      "name": "Config Name",
      "destinationName": "destination",
      "dataRepositoryType": "MSSharePoint",
      "includePaths": [
        "/path"
      ],
      "enumerationStatus": "IN_PROGRESS"
```

```
    }
  ]
}
```

Enumeration status values include:

- NEW
- IN_PROGRESS
- COMPLETE
- ERROR

1.4.3.1.1.1.2.1.3 Get a Metadata Configuration by ID

Retrieves the details of a specific metadata configuration by its ID.

Procedure

1. Send a GET request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations/<metadataConfigId>`.
   Include the ID of the metadata configuration that you want to retrieve in your path.
   Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/metadata/
configurations/<metadataConfigId> \
  --header 'AI-Resource-Group: <resource_group>' \
  --header 'Authorization: Bearer $TOKEN'
```

Results

Successful responses return code **200** and include details of your metadata configuration in JSON format.

Error responses include **404**: Metadata configuration not found

1.4.3.1.1.1.2.1.4 Delete a Metadata Configuration

Delete a specific metadata configuration by its ID.

Prerequisites

You've deleted any pipelines that are dependent on the metadata configuration that you want to delete.

Procedure

1. Send a DELETE request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations/<metadataConfigId>`.
   Include the ID of the metadata configuration that you want to retrieve in your path.
   Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Sample Code

```
curl --request DELETE \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/metadata/
configurations/<metadataConfigId> \
  --header 'AI-Resource-Group: <resource_group>' \
  --header 'Authorization: Bearer $TOKEN'
```

Results

Successful responses return code **202** and include a success message.

Error responses include:

- **404**: Metadata configuration not found
- **409**: Pipelines exist for this configuation
- **500**: Internal server error

1.4.3.1.1.1.2.1.5 List the Documents of a Metadata Configuration

Retrieves a paginated list of documents associated with a metadata configuration.

Procedure

1. Send a GET request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations/<metadataConfigId>/documents`.
   You can include optional parameters for pagination support. The following query parameters are supported:

Parameter	Type	Description
`absolutePath`	String	The file path to restrict the search to. Wildcards are supported.
`$top`	Integer	Max number of documents to return.
`$skip`	Integer	Number of documents to skip.
`$count`	Boolean	`true`: returns the total number of documents available on the server in the `documentsCount` field. `false`: returns the number of documents included in the current response in the `documentsCount` field.

   For example: `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations/<metadataConfigId>/documents?top=10&skip=0`

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/metadata/
configurations/<metadataConfigId>/documents \
  --header 'AI-Resource-Group: <resource_group>' \
```

```
  --header 'Authorization: Bearer $TOKEN'
```

Results

Successful responses return code **200** and include details of the documents associated with your metadata configurations in JSON format.

Output Code

```
{
  "count": 1,
  "resources": [
    {
      "id": "doc-uuid",
      "title": "Document Title",
      "absoluteFilePath": "/folder/file.pdf",
      "createdTimestamp": "2025-08-28T06:15:30Z",
      "type": "DOCUMENT",
      "metadata": []
    }
  ]
}
```

Type values include:

- FOLDER
- DOCUMENT

1.4.3.1.1.1.2.1.6 Get Details of a Document

Retrieves details of a document associated with a metadata configuration by document ID.

Procedure

1. Send a GET request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations/<metadataConfigId>/documents/<documentId>`.
   Include your metadata configuration ID and document ID in your request.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps

Header	Value
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/metadata/
configurations/<metadataConfigId>/documents/<documentId> \
  --header 'AI-Resource-Group: <resource_group>' \
  --header 'Authorization: Bearer $TOKEN'
```

Results

Successful responses return code **200** and include details of the document in JSON format.

Output Code

```
{
  "id": "doc-uuid",
  "title": "Document Title",
  "absoluteFilePath": "/folder/file.pdf",
  "createdTimestamp": "2025-08-28T06:15:30Z",
  "mimeType": "application/pdf",
  "fileSizeMb": "1.5",
  "metadata": []
}
```

Error responses include **404**:Document not found

1.4.3.1.1.1.2.1.7 Batch Update Document Metadata

Updates the metadata for multiple documents within a specific metadata configuration in a single request.

Context

The following limits apply:

- Maximum 1000 documents per request
- Maximum 10 metadata entries per document

Procedure

1. Send a PATCH request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations/<metadataConfigId>/documents`.
   Include your metadata configuration ID in your request.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Populate the sample code with the following values:

Field	Value
id	The ID of the document to be updated
labels	Your metadata labels in key-value pairs.

matchMode	One of the following modes:
	• ANY
	  • At least one metadata value matches.
	  • This is the default match mode.
	  • Defined only at injection time.
	• ALL
	  • All document metadata values must match exactly, or be a subset of the search metadata.
	  • Supported only at the document level.
	  • Defined only at injection time.

Sample Code

```
curl --request PATCH \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/metadata/
configurations/<metadataConfigId>/documents/<documentId> \
  --header 'AI-Resource-Group: <resource_group>' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer $TOKEN' \
  --data '{
  "value": [
    {
      "id": "<documentId>",
      "metadata": [
        {
          "key": "contentObjectId",
          "value": [
            "12345"
          ],
          "matchMode": "ANY"
        }
      ]
```

```
    }
  ]
}'
```

Results

Successful responses return code **200** and include details of the document in JSON format.

Output Code

```
{
"id": "document-uuid",
"metadata": [...]
}
```

1.4.3.1.1.1.2.2 Pipelines API

The Pipelines API allows you to provide unstructured documents from your data repositories, and create a document grounding pipeline for your resource group.

The API is proxied through the generative AI hub and incorporates vector stores, such as the managed HANA database.

Each pipeline represents a configured end-to-end process including the following steps:

1. Fetches documents from a supported data source.
2. Preprocesses and chunks the document content, and generates semantic embeddings. Semantic embeddings are multidimensional representations of textual information.
3. Stores semantic embeddings into the HANA Vector Store

Data Repositories

The following repositories and document types are supported:

- Microsoft SharePoint
- AWS S3
- SFTP
- SAP Build Work Zone
- SAP Document Management service
- Service Now
- Google Drive

Document Repository	Document (File) Format	Content Refresh	Document Content	Maximum Number of Documents
Microsoft SharePoint	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT, JSON	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline
AWS S3	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT, JSON	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline
SFTP	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline
SAP Build Work Zone	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline
SAP Document Management service	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline
Service Now	Knowledge Base Articles in HTML format	Daily	Plain Text	Maximum of 8000 Documents per Pipeline
Google Drive	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT,JSON	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline

API Request Format

When you build your API requests, follow the format outlined in the *Pipelines* section of the Grounding API. For more information, see Grounding API.

The endpoint for the pipelines API is `$AI_API_URL/v2/lm/document-grounding/pipelines`.

For region and pricing information, see SAP Note 3437766.

> ⚠ Caution
>
> Do not expose personally identifiable or privileged information in the documents exposed to your pipeline when using generative AI hub. Personally identifiable information is any data that can be used alone, or in combination, to identify the person that the data refers to. The document grounding capability has no mechanism for determining the type of data that it processes, and does not filter confidential or other privileged information.

Related Information

SAP Document Management Service

1.4.3.1.1.1.2.2.1 Create a Document Grounding Pipeline Using the Pipelines API

This API call creates a pipeline for indexing documents for a resource group.

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- You have created a generic secret. For more information, see Generic Secrets for Grounding [page 398].
- You've added your grounding documents to your document store and you're using a supported data source and document type. For more information, see Pipelines API [page 438].
- If you're using metadata, you've configured a metadata server. For more information, see Prepare your Metadata API Server [page 416].

Context

You can schedule automatic content updates by using a `cronExpression` when you create your pipeline. For more information, see Cron Expressions [page 463].

For repositories that support it, you can include metadata in your pipeline. Metadata is additional information associated with data that can help in identifying, categorizing, and retrieving relevant content.

> → Tip
>
> If you use the pipelines API, you do not need to call the Vector API separately. After the data is embedded, you can directly use the Retrieval API to query the vector store for relevant sections.

Procedure

You document grounding pipeline with your chosen repository. For more information, see the following:

- Create a Pipeline with Microsoft SharePoint [page 273]
- Create a Pipeline with AWS S3 [page 277]
- Create a Pipeline with SFTP [page 280]

- Create a Pipeline with SAP Build Work Zone [page 284]
- Create a Pipeline with SAP Document Management Service [page 458]
- Create a Document Grounding Pipeline Using the Pipelines API [page 440]

Next Steps

You can use API endpoints to view and manage the lifecycle of your data pipelines, executions and documents. For more information, see Data Pipelines [page 465], Executions [page 469] and Documents [page 471].

You can manually restart a pipeline. For more information, see Manually Restart a Document Grounding Pipeline [page 475].

1.4.3.1.1.1.2.2.1.1 Create a Pipeline with Google Drive

Creates a document grounding pipeline that indexes content from Google Drive using a generic secret, with optional metadata and scheduled updates.

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- The generic secret for Google Drive is available. For more information, see Create a Generic Secret for Google Drive [page 227].
- You have created a generic secret for Google Drive. For more information, see Generic Secrets for Grounding [page 398].
- You have added your grounding documents to Google Drive using a supported document type. For more information, see Pipelines API [page 438].
- If you are using metadata, you have configured a metadata server. For more information, see Prepare your Metadata API Server [page 416].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines`.

Without Metadata

The following shows an example of a Google Drive resource to be indexed.

Populate the sample code with the following values:

Field	Value
`{{resource_group}}`	The AI resource group assigned to your account
`$AI_API_URL`	The base URL of your SAP AI Core environment. This can also be set as an environment variable
`{{access_token}}`	Your access token for SAP AI Core
`<generic secret name>`	The name of the generic secret created for Google Drive
`<resource id>`	The ID of the shared drive or shared folder

Sample Code

```
curl --request POST \
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines" \
  --header "AI-Resource-Group: <resource-group>" \
  --header "Content-Type: application/json" \
  --header "Authorization: Bearer $TOKEN" \
  --data '{
    "type": "GoogleDrive",
    "configuration": {
      "destination": "<GENERIC_SECRET_NAME>",
      "googleDrive": {
        "resourceType": "<SHARED_DRIVE_OR_SHARED_FOLDER>",
        "resourceId": "<SHARED_DRIVE_ID_OR_SHARED_FOLDER_ID>",
        "includePaths": [
          "path/to/folder1",
          "path/to/folder2"
        ]
      }
    }
  }'
```

With Metadata

Metadata is additional information associated with data that can help in identifying, categorizing, and retrieving relevant content.

→ Tip

Alternatively, you can create a reusable metadata configurations and then reference your configuration when creating your pipeline. For more information, see Create a Metadata Configuration [page 428] and Create a Pipeline using a Metadata Configuration [page 462].

The metadata attribute is optional and must be used only if a metadata server is configured. To create a pipeline without metadata, see Create a Document Grounding Pipeline Using the Pipelines API [page 440].

Sample Code

```
curl --request POST \
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines" \
```

```
  --header "AI-Resource-Group: <resource-group>" \
  --header "Content-Type: application/json" \
  --header "Authorization: Bearer $TOKEN" \
  --data '{
    "type": "GoogleDrive",
    "configuration": {
      "destination": "<GENERIC_SECRET_NAME>",
      "googleDrive": {
        "resourceType": "<SHARED_DRIVE_OR_SHARED_FOLDER>",
        "resourceId": "<SHARED_DRIVE_ID_OR_SHARED_FOLDER_ID>"
      }
    },
    "metadata": {
      "destination": "<METADATA_SECRET_NAME>"
    }
  }'
```

With Scheduled Updates

`cronExpression` is optional. You can add it to your `configuration` to schedule pipeline updates at intervals of your choice.

Sample Code

```
curl --request POST \
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines" \
  --header "AI-Resource-Group: <resource-group>" \
  --header "Content-Type: application/json" \
  --header "Authorization: Bearer $TOKEN" \
  --data '{
    "type": "GoogleDrive",
    "configuration": {
      "destination": "<GEN>",
      "googleDrive": {
        "resourceType": "<SHARED_DRIVE_OR_SHARED_FOLDER>",
        "resourceId": "<SHARED_DRIVE_ID_OR_SHARED_FOLDER_ID>"
      },
      "cronExpression": "<CRON_EXPRESSION>"
    }
  }'
```

Pipeline Configuration Options

Google Drive pipelines support two resource types and an optional auto-discovery mode.

Parameters

- `resourceType` : Specifies the Google Drive resource type. Valid values are `SHARED_FOLDER` and `SHARED_DRIVE`.
- `resourceId`: Specifies the unique ID of the Google Drive folder or shared drive. If this parameter is omitted, auto-discovery mode is enabled and the pipeline scans all resources shared with the service account.

- `includePaths`: Specifies an array of strings to filter specific sub-folders. Example: `["ProjectDocs/Final"]`.

> **Note**
>
> If `resourceId` is not specified, the pipeline runs in auto-discovery mode and retrieves all files that the service account has permission to read.

Next Steps

After preparing your vectors, you can use the Retrieval API for chunks relevant to a query, or use the grounding module as part of an orchestration workflow for information retrieval and LLM interaction. For more information, see Retrieval API [page 504] or Using the Grounding Module [page 510].

You can add metadata to your pipeline, or update existing metadata by sending a patch request. For more information, see Update Metadata [page 464].

Supported File Types

The Google Drive pipeline supports common document and image file types. Google Workspace native formats are handled by remapping supported files to equivalent Microsoft Office formats. Unsupported file types are skipped during pipeline execution.

Supported file types:

- DOCX
- HTML (static)
- JPEG
- JPG
- JSON
- PDF
- PNG
- PPTX
- TIFF
- TXT

Google Drive-specific file types:

- Google Docs (`.gdoc`) are remapped to `.docx`.
- Google Slides (`.gslides`) are remapped to `.pptx`.

Unsupported file types:

- Other Google Workspace formats, such as Google Sheets
- Other file formats, such as `.xlsx`

1.4.3.1.1.1.2.2.1.2 Create a Pipeline with Microsoft SharePoint

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- You have created a generic secret. For more information, see Generic Secrets for Grounding [page 398].
- You've added your grounding documents to your document store and you're using a supported data source and document type. For more information, see Pipelines API [page 438].
- If you're using metadata, you've configured a metadata server. For more information, see Prepare your Metadata API Server [page 416].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines`.

Without Metadata

The following shows an example of a SharePoint site to be indexed.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name>	The name of the generic secret created for MS SharePoint
<site name>	The name of the SharePoint site to be indexed
["/<folder1>", "/<folder2>/<sub-folder1>"]	An array of folders within the SharePoint site for selective indexing (Optional)

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
```

```
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "MSSharePoint",
  "configuration": {
    "destination": "<generic secret name>",
    "sharePoint": {
      "site": {
        "name": "<site name>",
        "includePaths": ["/<folder1>", "/<folder2>/<sub-folder1>"]
      }
    }
  }
}'
```

> **Note**
>
> The `includePaths` parameter is optional. It takes an array of folder paths within the SharePoint site to be indexed and limits the scope of the pipeline to the specified sub folders or files, meaning that only this content is indexed.
>
> The `includePaths` parameter input is restricted to folder paths within the default document folder of a SharePoint site. No other drives, document library paths, or formats are supported.

With Metadata

Metadata is additional information associated with data that can help in identifying, categorizing, and retrieving relevant content.

> **→ Tip**
>
> Alternatively, you can create a reusable metadata configurations and then reference your configuration when creating your pipeline. For more information, see Create a Metadata Configuration [page 428] and Create a Pipeline using a Metadata Configuration [page 462].

The metadata attribute is optional. It accepts the destination name, which is used to connect to the Microsoft SharePoint metadata server to retrieve metadata for document indexing.

The metadata attribute should be used only if a metadata server is configured. To create a grounding pipeline without metadata, see Create a Document Grounding Pipeline Using the Pipelines API [page 440].

The following shows an example of a SharePoint site to be indexed with metadata.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core

Field	Value
<generic secret name> (destination)	The name of the generic secret created for Microsoft SharePoint
<site name>	The name of the SharePoint site to be indexed
<generic secret name> (metadata)	The name of the generic secret created for Microsoft SharePoint Metadata Server

⚠ Restriction

`metadata` and `includePaths` cannot be used together in a single payload request.

{·} Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "MSSharePoint",
  "configuration": {
    "destination": "<generic secret name>",
    "sharePoint": {
      "site": {
        "name": "<site name>"
      }
    }
  },
  "metadata": {
    "destination": "<generic secret name>"
  }
}'
```

You can search a pipeline later in your workflow by using the `dataRepositoryMetadata` attribute in the `metadata` field. Metadata organizes information using categories (keys) paired with their related values. For example:

```
...
    "metadata": {
        "dataRepositoryMetadata": [
            {
                "key": "example",
                "value": [
                    "demo"
                ]
            }
        ]
    }
...
```

For more information, see Search a Pipeline [page 476].

With Scheduled Updates

`cronExpression` is optional. You can add it to your `configuration` to schedule updates to your pipeline, at intervals of your choice.

The following shows an example of a SharePoint site to be indexed at regular intervals.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name>	The name of the generic secret created for MS SharePoint
<site name>	The name of the SharePoint site to be indexed
cronExpression	The cronExpression for the update frequency of your choice (optional). For more information, see Cron Expressions [page 463].

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "MSSharePoint",
  "configuration": {
    "destination": "<generic secret name>",
    "cronExpression":"<cron expression>"
    "sharePoint": {
      "site": {
        "name": "<site name>"
      }
    },
  }
}'
```

Note

Each repetition of your pipeline is a new deployment, and incurs costs.

Next Steps

After preparing your vectors, you can use the Retrieval API for chunks relevant to a query, or use the grounding module as part of an orchestration workflow for information retrieval and LLM interaction. For more information, see Retrieval API [page 504] or Using the Grounding Module [page 510].

You can add metadata to your pipeline, or update existing metadata by sending a patch request. For more information, see Update Metadata [page 464].

1.4.3.1.1.1.2.2.1.3 Create a Pipeline with AWS S3

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- You have created a generic secret. For more information, see Generic Secrets for Grounding [page 398].
- You've added your grounding documents to your document store and you're using a supported data source and document type. For more information, see Pipelines API [page 438].
- If you're using metadata, you've configured a metadata server. For more information, see Prepare your Metadata API Server [page 416].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines`.

Without Metadata

The following shows an example of an AWS S3 repository to be indexed.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	{{access_token}}
<generic secret name>	Name of the generic secret created for AWS S3

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
```

```
  --data '{
  "type": "S3",
  "configuration": {
    "destination": "<generic secret name>",
    "s3": {
        "includePaths": ["/Sample Files/folder1"]
    }
  }
}'
```

With Metadata

Metadata is additional information associated with data that can help in identifying, categorizing, and retrieving relevant content.

> → Tip
>
> Alternatively, you can create a reusable metadata configurations and then reference your configuration when creating your pipeline. For more information, see Create a Metadata Configuration [page 428] and Create a Pipeline using a Metadata Configuration [page 462].

The metadata attribute is optional. It accepts the destination name, which is used to connect to the Microsoft SharePoint metadata server to retrieve metadata for document indexing.

The metadata attribute should be used only if a metadata server is configured. To create a grounding pipeline without metadata, see Create a Document Grounding Pipeline Using the Pipelines API [page 440].

The following shows an example of an AWS S3 repository to be indexed with metadata:

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name> (configuation)	The name of the generic secret created for AWS S3
<generic secret name> (metadata)	The name of the generic secret created for the AWS S3 Metadata Server

> ⚠ Restriction
>
> `metadata` and `includePaths` cannot be used together in a single payload request.

> {·} Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
```

```
   --data '{
   "type": "S3",
   "configuration": {
    "destination": "<generic secret name>",
     "s3": {
         "includePaths": ["/Sample Files/folder1"]
     }
   },
   "metadata": {
     "destination": "<generic secret name>"
   }
 }'
```

You can search a pipeline later in your workflow by using the `dataRepositoryMetadata` attribute in the `metadata` field. Metadata organizes information using categories (keys) paired with their related values. For example:

```
...
    "metadata": {
        "dataRepositoryMetadata": [
            {
                "key": "example",
                "value": [
                    "demo"
                ]
            }
        ]
    }
...
```

For more information, see Search a Pipeline [page 476].

With Scheduled Updates

`cronExpression` is optional. You can add it to your `configuration` to schedule updates to your pipeline, at intervals of your choice.

The following shows an example of an AWS S3 repository to be indexed at regular intervals.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name>	Name of the generic secret created for AWS S3
cronExpression	The cronExpression for the update frequency of your choice (optional). For more information, see Cron Expressions [page 463].

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
    "type": "S3",
    "configuration": {
        "destination": "<generic secret name>",
        "cronExpression": "<cron expression>"
        "s3": {
            "includePaths": [
                "/Sample Files/folder1"
            ],
        }
    }
}'
```

Note

Each repetition of your pipeline is a new deployment, and incurs costs.

Next Steps

After preparing your vectors, you can use the Retrieval API for chunks relevant to a query, or use the grounding module as part of an orchestration workflow for information retrieval and LLM interaction. For more information, see Retrieval API [page 504] or Using the Grounding Module [page 510].

You can add metadata to your pipeline, or update existing metadata by sending a patch request. For more information, see Update Metadata [page 464].

1.4.3.1.1.1.2.2.1.4 Create a Pipeline with SFTP

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- You have created a generic secret. For more information, see Generic Secrets for Grounding [page 398].
- You've added your grounding documents to your document store and you're using a supported data source and document type. For more information, see Pipelines API [page 438].
- If you're using metadata, you've configured a metadata server. For more information, see Prepare your Metadata API Server [page 416].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines`.

Without Metadata

The following shows an example of an SFTP repository to be indexed:

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name>	The name of the generic secret created for SFTP

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "SFTP",
  "configuration": {
    "destination": "<generic secret name>",
    "sftp": {
        "includePaths": ["/Sample Files/folder1"]
    }
  }
}'
```

With Metadata

Metadata is additional information associated with data that can help in identifying, categorizing, and retrieving relevant content.

→ Tip

Alternatively, you can create a reusable metadata configurations and then reference your configuration when creating your pipeline. For more information, see Create a Metadata Configuration [page 428] and Create a Pipeline using a Metadata Configuration [page 462].

The metadata attribute is optional. It accepts the destination name, which is used to connect to the Microsoft SharePoint metadata server to retrieve metadata for document indexing.

The metadata attribute should be used only if a metadata server is configured. To create a grounding pipeline without metadata, see Create a Document Grounding Pipeline Using the Pipelines API [page 440].

The following shows an example of an SFTP repository to be indexed with metadata:

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name> (configuation)	The name of the generic secret created for SFTP
<generic secret name> (metadata)	The name of the generic secret created for the SFTP Metadata Server

⚠ Restriction

`metadata` and `includePaths` cannot be used together in a single payload request.

{·} Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "SFTP",
  "configuration": {
    "destination": "<generic secret name>",
    "sftp": {
        "includePaths": ["/Sample Files/folder1"]
    },
  "metadata": {
    "destination": "<generic secret name>"
  }
}'
```

The metadata attribute is optional. It accepts the destination name, which is used to connect to the SFTP metadata server to retrieve metadata for document indexing.

You can search a pipeline later in your workflow by using the `dataRepositoryMetadata` attribute in the `metadata` field. Metadata organizes information using categories (keys) paired with their related values. For example:

```
...
    "metadata": {
        "dataRepositoryMetadata": [
            {
                "key": "example",
                "value": [
                    "demo"
                ]
            }
        ]
```

```
    }
...
```

For more information, see Search a Pipeline [page 476].

With Scheduled Updates

`cronExpression` is optional. You can add it to your `configuration` to schedule updates to your pipeline, at intervals of your choice.

The following shows an example of an SFTP repository to be indexed at regular intervals.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name>	The name of the generic secret created for SFTP
cronExpression	The cronExpression for the update frequency of your choice (optional). For more information, see Cron Expressions [page 463].

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
    "type": "SFTP",
    "configuration": {
        "destination": "<generic secret name>",
        "cronExpression": "<cron expression>",
        "sftp": {
            "includePaths": [
                "/Sample Files/folder1"
            ]
        }
    }
}'
```

Note

Each repetition of your pipeline is a new deployment, and incurs costs.

Next Steps

After preparing your vectors, you can use the Retrieval API for chunks relevant to a query, or use the grounding module as part of an orchestration workflow for information retrieval and LLM interaction. For more information, see Retrieval API [page 504] or Using the Grounding Module [page 510].

You can add metadata to your pipeline, or update existing metadata by sending a patch request. For more information, see Update Metadata [page 464].

1.4.3.1.1.1.2.2.1.5 Create a Pipeline with SAP Build Work Zone

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- You have created a generic secret. For more information, see Generic Secrets for Grounding [page 398].
- You've added your grounding documents to your document store and you're using a supported data source and document type. For more information, see Pipelines API [page 438].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines`.

The following shows an example of a SAP Build Work Zone site to be indexed.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name> (destination)	The name of the generic secret created for SAP Build Work Zone

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
```

```
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "WorkZone",
  "metadata": {
    "destination": "<generic secret name>"
  }
}'
```

Scheduling updates

Optionally, you can add a `configuration` containing a `cronExpression` to schedule updates to your pipeline, at intervals of your choice.

For more information, see Cron Expressions [page 463].

For example:

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
    "type": "WorkZone",
    "metadata": {
        "destination": "<generic secret name>"
    },
    "configuration": {
        "cronExpression": "<cron expression>"
    }
}'
```

> **Note**
>
> Each repetition of your pipeline is a new deployment, and incurs costs.

Next Steps

After preparing your vectors, you can use the Retrieval API for chunks relevant to a query, or use the grounding module as part of an orchestration workflow for information retrieval and LLM interaction. For more information, see Retrieval API [page 504] or Using the Grounding Module [page 510].

Related Information

Integration With Document Grounding SAP Build Work Zone, Advanced Edition

1.4.3.1.1.1.2.2.1.6 Create a Pipeline with SAP Document Management Service

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- You have created a generic secret. For more information, see Generic Secrets for Grounding [page 398].
- You've added your grounding documents to your document store and you're using a supported data source and document type. For more information, see Pipelines API [page 438].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines`.

The following shows an example of a SAP Document Management service site to be indexed.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name> (destination)	The name of the generic secret created for SAP Document Management service

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "SDM",
  "metadata": {
    "destination": "<generic secret name>"
  }
}'
```

Scheduling updates

Optionally, you can add a `configuration` containing a `cronExpression` to schedule updates to your pipeline, at intervals of your choice.

For more information, see Cron Expressions [page 463].

For example:

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
    "type": "SDM",
    "metadata": {
        "destination": "sdm-secret-aicore"
    },
    "configuration": {
        "cronExpression": "<cron expression>"
    }
}'
```

> **Note**
>
> Each repetition of your pipeline is a new deployment, and incurs costs.

Next Steps

After preparing your vectors, you can use the Retrieval API for chunks relevant to a query, or use the grounding module as part of an orchestration workflow for information retrieval and LLM interaction. For more information, see Retrieval API [page 504] or Using the Grounding Module [page 510].

1.4.3.1.1.1.2.2.1.7 Create a Pipeline with ServiceNow

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- You have created a generic secret. For more information, see Generic Secrets for Grounding [page 398].
- You've added your grounding documents to your document store and you're using a supported data source and document type. For more information, see Pipelines API [page 438].
- If you're using metadata, you've configured a metadata server. For more information, see Prepare your Metadata API Server [page 416].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines`.

Without Metadata

The following shows an example of a ServiceNow site to be indexed.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name>	The name of the generic secret created for ServiceNow

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "ServiceNow",
  "configuration": {
    "destination": "<GENERIC_SECRET_NAME>"
  }
}'
```

With Metadata

Metadata is additional information associated with data that can help in identifying, categorizing, and retrieving relevant content.

The metadata attribute is optional. It accepts the destination name, which is used to connect to the ServiceNow metadata server to retrieve metadata for document indexing.

The metadata attribute should be used only if a metadata server is configured. To create a grounding pipeline without metadata, see Create a Document Grounding Pipeline Using the Pipelines API [page 440].

The following shows an example of a ServiceNow site to be indexed with metadata.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name> (destination)	The name of the generic secret created for ServiceNow
<generic secret name> (metadata)	The name of the generic secret created for the ServiceNow Metadata Server

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "ServiceNow",
  "configuration": {
    "destination": "<GENERIC_SECRET_NAME>"
  },
  "metadata": {
    "destination": "<METADATA_SECRET_NAME>"
  }
}'
```

With Scheduled Updates

`cronExpression` is optional. You can add it to your `configuration` to schedule updates to your pipeline, at intervals of your choice.

The following shows an example of a ServiceNow site to be indexed at regular intervals.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name>	The name of the generic secret created for ServiceNow
cronExpression	The cronExpression for the update frequency of your choice (optional). For more information, see Cron Expressions [page 463].

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "ServiceNow",
  "configuration": {
    "destination": "<GENERIC_SECRET_NAME>",
    "cronExpression": "<CRON_EXPRESSION>"
  }
}'
```

Note

Each repetition of your pipeline is a new deployment, and incurs costs.

Next Steps

After preparing your vectors, you can use the Retrieval API for chunks relevant to a query, or use the grounding module as part of an orchestration workflow for information retrieval and LLM interaction. For more information, see Retrieval API [page 504] or Using the Grounding Module [page 510].

1.4.3.1.1.1.2.2.1.8 Create a Pipeline using a Metadata Configuration

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- You have created a generic secret. For more information, see Generic Secrets for Grounding [page 398].
- You've created a metadata configuration and have noted the metadata configuration ID. For more information, see Create a Metadata Configuration [page 428] and List Metadata Configurations [page 430].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines`.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Include the ID of your metadata configuration in your request. For example:

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: <resource group>' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer $TOKEN'
  --data '{
    "type": "metadata",
    "configuration": {
      "metadataConfigId": "<metadata config id>"
     }
}'
```

1.4.3.1.1.1.2.2.1.9 Cron Expressions

You can schedule updates to your pipeline, at intervals of your choice by adding a `cronExpression` when you create your pipeline.

Cron expressions have five fields, represented by a string of asterisks. The values for these fields are as follows:

Pipeline Scheduling: Fields and Values

*	*	*	*	*
Minute (0–59)	Hour (0–23)	Day of the month (1–31)	Month (1–12)	Day of the week (0–6)

The minimum interval supported in the pipeline API is 1 hour.

Pipeline Scheduling Fields: Example Values

Expression	Schedule
0****	Every hour
00***	Every day at 12:00 a.m.
00**FRI	At 12:00 a.m. only on Friday
001**	At 12:00 a.m. on the first day of the month

> **Note**
>
> Each repetition of your pipeline is a new deployment, and incurs costs.

1.4.3.1.1.1.2.2.1.10 Update Metadata

Prerequisites

- You have created a pipeline using a repository that supports metadata. For more information, see Create a Document Grounding Pipeline Using the Pipelines API [page 440].

Context

Metadata is additional information associated with data that can help in identifying, categorizing, and retrieving relevant content. You can use your metadata values later to search your pipeline. For more information, see Search a Pipeline [page 476].

Procedure

For repository types that support metadata, you can add or update your metadata by sending a PATCH request to to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}`.

Include the pipeline ID for the pipeline that you want to update.

Ensure that you've the following headers set:

Header	Value
Authorization	Bearer $AUTH_TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. This URL can also be set as an environment variable.

Metadata organizes information using categories (keys) paired with their related values.

> **Sample Code**

```
curl --request PATCH \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}} \
  --header 'ai-resource-group: <resource_group>' \
  --header 'authorization: Bearer $AUTH_TOKEN' \
```

```
  --header 'content-type: application/json' \
  --data '{
  "metadata": {
    "dataRepositoryMetadata": [
      {
        "key": "pipeline/purpose",
        "value": [
          "patch-pipeline1"
        ]
      },
      {
        "key": "pipeline/owner",
        "value": [
          "data-engineering",
          "analytics-team"
        ]
      }
    ]
  }
}'
```

1.4.3.1.1.1.2.2.2 Data Pipelines

You can get the details of your data pipelines using the following endpoints:

Get All Pipelines

Use this endpoint to retrieve all document grounding pipelines created in your specified AI resource group.

Each pipeline represents a configured end-to-end process that:

- Fetches documents from a supported data source
- Preprocesses and chunks the content
- Stores semantic embeddings into the HANA Vector Store

This is useful when you want to:

- Review existing pipelines before creating or executing a new one
- Manage pipeline inventory programmatically
- Support dashboards, monitoring, or integration flows
- Enable pipeline reusability across projects

> **Note**
>
> If you use the Pipeline APIs for creating and retrieving Pipeline details, you do not need to use the Vector API separately, it's already handled as part of the pipeline execution. After a pipeline runs, you can use the Retrieval API to query grounded content.
>
> The endpoint for retrieving document pipelines is `$AI_API_URL/v2/lm/document-grounding/pipelines`

You can use the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines` to retrieve all document grounding pipelines.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines\
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

This endpoint returns a response containing an array of pipeline objects, in JSON format.

Available statuses for pipelines are as follows:

- NEW
- UNKNOWN
- INPROGRESS
- FINISHED
- FINISHEDWITHERRORS
- TIMEOUT

Pagination Support

You can use OData-compliant query parameters to manage large result sets:

Parameter	Type	Description
$top	Integer	Limits the number of results returned. for example., $top=10 returns 10 results.
$skip	Integer	Skips a specific number of results. for example, $skip=20 skips the first 20.
$count	Boolean	If true, includes the total count in the response. If false, includes the total number of records in the response.
metaConfigId	String	Filter pipelines by metadata configuration ID.

For example:

Sample Code

```
curl --request GET
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines?
$top=10&$skip=20&$count=true"
  --header "AI-Resource-Group: {{resource_group}}"
  --header 'Authorization: Bearer {{access_token}}'
```

or:

```
curl --request GET
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines?
metadataConfigId=<yourMetadataConfigId>"
  --header "AI-Resource-Group: {{resource_group}}"
  --header 'Authorization: Bearer {{access_token}}'
```

Successful responses include response code **200** and a filtered list of pipelines.

Get Pipeline Details

This endpoint lets you fetch full details of a specific document grounding pipeline by using its unique pipelineId.

It is useful when you want to:

- Inspect the pipeline's configuration (fore example, data source, chunking strategy, schedule)
- Debug or validate how a pipeline is set up
- Display pipeline metadata
- Get last updated time and status before reusing or modifying a pipeline

You can get pipeline details using the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}`.

Populate the code snippet with the following:

- `{{resource_group}}`: the ID of the specific pipeline you want to retrieve.
- `AI_API_URL`: the base URL of your SAP AI Core environment.
- `{{access_token}}`: Your access token for SAP AI Core
- `{{pipelineId}}`: the AI resource group ID assigned to your account.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}} \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

This endpoint returns a response containing details of the pipeline configuration and metadata, in JSON format.

Available statuses for pipelines are as follows:

- NEW

- UNKNOWN
- INPROGRESS
- FINISHED
- FINISHEDWITHERRORS
- TIMEOUT

Get Pipeline Status

This endpoint allows you to retrieve the current status of a specific document grounding pipeline. It provides information on the lifecycle state, readiness, or any ongoing processing of the pipeline.

Populate the code snippet with the following:

- `{{resource_group}}`: the ID of the specific pipeline you want to retrieve..
- `AI_API_URL`: the base URL of your SAP AI Core environment.
- `{{access_token}}`: Your access token for SAP AI Core
- `{{pipelineId}}`: the AI resource group ID assigned to your account.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/status \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

This endpoint returns a response containing details of the pipeline status, in JSON format.

Available statuses for pipelines are as follows:

- NEW
- UNKNOWN
- INPROGRESS
- FINISHED
- FINISHEDWITHERRORS
- TIMEOUT

Delete a Pipeline

Use this endpoint to permanently delete a specific document grounding pipeline by its unique pipelineId. This is typically done when a pipeline is no longer required, misconfigured, or needs to be recreated with different settings. After deletion, the pipeline and its configuration are irreversibly removed.

You can delete a pipeline using the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}`.

Populate the code snippet with the following:

- `{{resource_group}}`: the ID of the specific pipeline you want to retrieve..
- `AI_API_URL`: the base URL of your SAP AI Core environment.
- `{{access_token}}`: Your access token for SAP AI Core
- `{{pipelineId}}`: the AI resource group ID assigned to your account.

Sample Code

```
curl --request DELETE \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}} \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

This endpoint returns a response containing confirmation of deletion, in JSON format.

1.4.3.1.1.1.2.2.3 Executions

You can get the details of your executions in data pipelines using the following endpoints:

List All Executions of a Document Grounding Pipeline

This endpoint allows you to list all executions associated with a specific document grounding pipeline. The response includes execution metadata such as status, timestamps, and identifiers.

You can list all the executions by using the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/executions?lastExecution=false`.

Populate the code snippet with the following:

- `{{pipelineId}}`: The ID of the pipeline whose executions you want to list
- `lastExecution`: Set to `true` to fetch only the latest execution (the default value is `false`)
- `{{resource_group}}`: The AI resource group ID assigned to your account
- `{{access_token}}`: Your access token for SAP AI Core

This endpoint returns a response in JSON format.

Sample Code

```
curl --request GET \
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/
executions?lastExecution=false" \
  --header "AI-Resource-Group: {{resource_group}}"
  --header 'Authorization: Bearer {{access_token}}'
```

Pagination Support

You can use OData-compliant query parameters to manage large result sets:

Parameter	Type	Description
`$top`	Integer	Limits the number of results returned. For example, `$top=10` returns 10 results.
`$skip`	Integer	Skips a specific number of results. For example, `$skip=20` skips the first 20.
`$count`	Boolean	If `true`, includes the total count in the response. If `false`, includes the total number of records in the response.

Example

This code returns a JSON array of pipeline execution objects.

```
curl --request GET
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/
executions?$top=10&$skip=20&$count=true"
  --header "AI-Resource-Group: {{resource_group}}"
  --header 'Authorization: Bearer {{access_token}}'
```

Available statuses for executions are as follows:

- NEW
- UNKNOWN
- INPROGRESS
- FINISHED
- FINISHEDWITHERRORS
- TIMEOUT

Get Specific Execution Details

This endpoint allows you to retrieve the details of a specific execution of a document grounding pipeline by providing the pipeline ID and execution ID.

You can retrieve a specific execution by using the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/executions/{{executionId}}`.

Populate the code snippet with the following:

- `{{pipelineId}}`: The ID of the pipeline that contains the execution that you want to retrieve
- `{{executionId}}`: The ID of the specific execution
- `{{resource_group}}`: The AI resource group ID assigned to your account
- `{{access_token}}`: Your access token for SAP AI Core

This endpoint returns a response in JSON format. The response includes fields such as `status`, `startTime`, `endTime`, and any `errorMessage`.

Sample Code

```
curl --request GET \
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/
executions/{{executionId}}" \
  --header "AI-Resource-Group: {{resource_group}}"
  --header 'Authorization: Bearer {{access_token}}'
```

Available statuses for executions are as follows:

- NEW
- UNKNOWN
- INPROGRESS
- FINISHED
- FINISHEDWITHERRORS
- TIMEOUT

1.4.3.1.1.1.2.2.4 Documents

You can get the details of your documents in data pipelines using the following endpoints:

List All Documents for an Execution

This endpoint allows you to retrieve all documents associated with a specific execution, including the processing statuses of the documents. You must provide the pipeline ID and the execution ID as parameters.

You can list all documents for an execution by using the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/executions/{{executionId}}/documents`.

Populate the code snippet with the following:

- `{{pipelineId}}`: The pipeline ID
- `{{executionId}}`: The ID of the specific execution
- `{{resource_group}}`: The AI resource group ID assigned to your account

This endpoint returns a response in JSON format. The response includes the list of all documents processed during the specified execution.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/
executions/{{executionId}}/documents \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

Available statuses for documents are as follows:

- TO_BE_PROCESSED
- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED_TO_BE_RETRIED
- TO_BE_SCHEDULED

Pagination Support

You can use OData-compliant query parameters to manage large result sets:

Parameter	Type	Description
`$top`	Integer	Limits the number of results returned. For example, `$top=10` returns 10 results.
`$skip`	Integer	Skips a specific number of results. For example, `$skip=20` skips the first 20.
`$count`	Boolean	If `true`, includes the total count in the response. If `false`, includes the total number of records in the response.

Example

This code returns a JSON array of document objects associated with the pipeline execution.

```
curl --request GET
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/
executions{{executionId}}/documents?$top=10&$skip=20&$count=true"
  --header "AI-Resource-Group: {{resource_group}}" \
  --header 'Authorization: Bearer {{access_token}}'
```

Available statuses for documents are as follows:

- TO_BE_PROCESSED
- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED_TO_BE_RETRIED
- TO_BE_SCHEDULED

Get a Specific Document from an Execution

This endpoint allows you to retrieve detailed information for a specific document within an execution, including its processing status. The endpoint requires the input parameters for the pipeline ID, execution ID, and document ID.

You can retrieve a specific document by using the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/executions/{{executionId}}/documents/{{documentID}}`.

Populate the code snippet with the following:

- `{{pipelineId}}`: The pipeline ID
- `{{executionId}}`: The ID of the specific execution
- `{{documentId}}`: The ID of the specific document
- `{{resource_group}}`: The AI resource group ID assigned to your account
- `{{access_token}}`: Your access token for SAP AI Core

This endpoint returns a response in JSON format. The response includes the specific document processed during the specified execution. It also includes the document's processing status by pipeline ID, execution ID, and document ID.

Sample Code

```
curl --request GET \
  --url $AI_API_URL /v2/lm/document-grounding/pipelines/{{pipelineId}}/
executions/{{executionId}}/documents/{{documentID}} \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

Available statuses for documents are as follows:

- TO_BE_PROCESSED
- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED_TO_BE_RETRIED
- TO_BE_SCHEDULED

List All Documents for a Pipeline

This endpoint allows you to retrieve all documents for a pipeline, including the processing statuses by pipeline ID.

You can retrieve all documents by using the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/documents`.

Populate the code snippet with the following:

- `{{pipelineId}}`: The pipeline ID

- `{{resource_group}}`: The AI resource group ID assigned to your account
- `{{access_token}}`: Your access token for SAP AI Core

This endpoint returns a response in JSON format.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/
documents \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

Available statuses for documents are as follows:

- TO_BE_PROCESSED
- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED_TO_BE_RETRIED
- TO_BE_SCHEDULED

Pagination Support

You can use OData-compliant query parameters to manage large result sets:

Parameter	Type	Description
`$top`	Integer	Limits the number of results returned. For example, `$top=10` returns 10 results.
`$skip`	Integer	Skips a specific number of results. For example, `$skip=20` skips the first 20.
`$count`	Boolean	If `true`, includes the total count in the response. If `false`, includes the total number of records in the response.

Example

This code returns a JSON array of document objects associated with the pipeline execution.

```
curl --request GET
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/
documents?$top=10&$skip=20&$count=true"
  --header "AI-Resource-Group: {{resource_group}}"
  --header 'Authorization: Bearer {{access_token}}'
```

Get a Specific Document for a Pipeline

This endpoint allows you to retrieve detailed information about a document within a specific pipeline, including the document's processing status. The endpoint requires the pipeline ID and document ID as parameters.

Populate the code snippet with the following:

- `{{pipelineId}}`: The pipeline ID
- `{{documentId}}`: The ID of the specific document
- `{{resource_group}}`: The AI resource group ID assigned to your account
- `{{access_token}}`: Your access token for SAP AI Core

This endpoint returns a response in JSON format. The response includes the specific document processed in the pipeline.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/
documents/{{documentID}}
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

Available statuses for documents are as follows:

- TO_BE_PROCESSED
- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED_TO_BE_RETRIED
- TO_BE_SCHEDULED

1.4.3.1.1.1.2.2.5 Manually Restart a Document Grounding Pipeline

This endpoint allows you to retrain an existing pipeline manually.

Context

You can start an existing pipeline by sending a POST request to the endpoint: `/v2/lm/document-grounding/pipelines/trigger` and including your pipeline ID in your request.

Procedure

1. Populate the sample code with the following values:

	Value
`{{resource_group}}`	The AI resource group assigned to your account
`$AI_API_URL`	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
`{{access_token}}`	Your access token for SAP AI Core
`{{pipelineId}}`	The ID of the pipeline that you want to start
`metadataOnly`	A Boolean value. `true` updates the metadata only. `false` updates you semantic embeddings by processing the documents in your repository.

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/trigger \
  --header 'ai-resource-group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
  --header 'content-type: application/json' \
  --data '{
    "pipelineId":"{{pipelineId}}",
    "metadataOnly": "<Boolean>"
  }'
```

Results

The response includes a response code. Code 202 is a successful response.

1.4.3.1.1.1.2.2.6 Search a Pipeline

Prerequisites

You've created a document grounding pipeline, and included your choice of metadata in the `dataRepositoryMetadata` field. For more information, see Create a Document Grounding Pipeline Using the Pipelines API [page 440].

Procedure

Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/search`. Include your data repository key:value pair in your request.

Set the following in the headers:

- `AI_API_URL`: the base URL of your SAP AI Core environment.
- `{{resource_group}}`: The AI resource group ID or document grounding management
- `{{access_token}}`: Your access token for SAP AI Core

For example:

```
curl --request POST \
--url $AI_API_URL/v2/lm/document-grounding/pipelines/search \
--header 'AI-Resource-Group: {{resource_group}}'
--data '{
    "dataRepositoryMetadata": [
        {
            "key": "example",
            "value": [
                "demo"
            ]
        }
    ]
}'
```

The output lists any pipelines that match the data repository metadata.

Output Code

```
{
    "count": 1,
    "resources": [
        {
            "id": "<pipelineid>",
            "status": "<status-of-pipeline-execution>",
            "type": "<type-of-repository>",
            "configuration": {
                "sharePoint": {
                    "id": "<site-id>",
                    "name": "<site-name>"
                }
            }
        }
    ]
}
```

1.4.3.1.1.1.2.3 Vector API

Vector API is a set of REST APIs used for document ingestion and retrieval using vector embeddings.

You can use the Vector API to:

- Create collections.
- Create documents by directly using chunks of data.

- Store data in a vector database.
- Provide repository IDs to access the data.

The Vector API incorporates the following units:

Collections: A logical container used to store and manage embedded documents.

Documents: A unit of text-based content that you upload into a collection, to be used for semantic search and grounding.

Chunks: A small segment of a document's text, used in the Vector API to improve the accuracy and efficiency of semantic search.

Capacity Quotas and Input Limits

Vector Ingestion – Capacity Limits

The following capacity limits apply:

Category	Description	Limit Type	Limit
Collections	Max collections	Quota	55,000
Documents	Max documents per collection	Quota	100,000
Documents	Max documents per ingestion request	Input Limit	100
Chunks	Max chunks per document	Quota	200,000

Metadata Limits (Ingestion and Query Time)

The following metadata limits apply:

Metadata Type	Limit Type	Limit
Collection Metadata	Quota / Input Limit	500 / 5,000
Document Metadata	Quota / Input Limit	500 / 5,000
Chunk Metadata	Quota / Input Limit	500 / 5,000

Vector Search – Query Limits

The following query limits apply:

Category	Description	Limit Type	Limit
Filters	Max filters per query	Input Limit	15
Collections	Max collection IDs per query	Input Limit	5,000
Results	Max chunk results per query	Input Limit	100
Results	Max document results per query	Input Limit	10

1.4.3.1.1.1.2.3.1 Preparing Data Using the Vector API

Vector API is a microservice provided with a Rest API and endpoints for creating and managing collection and documents.

Prerequisites

You have created a resource group for grounding purposes. For more information, see Create a Resource Group for Grounding [page 397].

To insert your document chunks into the vector store, you will need to follow the following operations:

1. Create a collection For more information see Create a Collection [page 480].
2. Insert documents and chunks into the collection. For more information see Create a Document [page 487].

API Request Format

When you build your API requests, follow the format outlined in the *Vector* section of the Grounding API. For more information, see Grounding API.

> ⚠ Caution
>
> Do not expose personally identifiable or privileged information in your documents or chunks when using generative AI hub. Personally identifiable information is any data that can be used alone, or in combination, to identify the person that the data refers to. The document grounding capability has no mechanism for determining the type of data that it processes, and does not filter confidential or other privileged information.

Next Steps

After preparing your vectors, you can use the Retrieval API for chunks relevant to a query, or use the grounding module as part of an orchestration workflow for information retrieval and LLM interaction. For more information, see Retrieval API [page 504] or Using the Grounding Module [page 510].

Alternatively, you can search specific collections using vector search. For more information, see Vector Search [page 498]

1.4.3.1.1.1.2.3.1.1 Collections

1.4.3.1.1.1.2.3.1.1.1 Create a Collection

This endpoint allows you to create a new collection. Collections are logical containers used to store and manage embedded documents and their associated chunks.

Prerequisites

You have created a resource group for grounding purposes. For more information, see Create a Resource Group for Grounding [page 397]

Context

This endpoint allows you to create a new collection for storing documents and their associated chunks.

Procedure

1. Send a POST request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections`.
   Populate the sample code with the following values:

Placeholder	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<title of the collection>	Your choice of title for the collection
<your choice of valid uuid>	Your choice of UUID that can be used later to locate the collection
<embedding-model-name>	The name of the embedding model to be used during vectorization
metadata.purpose	Your choice of text about the purpose of the collection
metadata <key> <value> pair	Your choice of metadata key value pair

> **Note**
>
> The `id`, `title` and `metadata` fields are optional.

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "id": "<your choice of valid uuid>",
  "title": "<title of the collection>",
  "embeddingConfig": {
    "modelName": "<embedding-model-name>"
  },
  "metadata": [
    {
      "key": "purpose",
      "value": [
        "<text>"
      ]
    },
    {
      "key": "<a-random-key>",
      "value": [
        "<text>"
      ]
    }
  ]
}'
```

Results

The response is returned in JSON format, and includes a `location` header that contains the endpoint that can be used to get the status of collection creation.

Next Steps

Check the status of your collection. For more information, see Get Collection Creation Status [page 484].

1.4.3.1.1.1.2.3.1.1.2 Get Collection Details

This endpoint allows you to retrieve the details of a specific collection using its collection ID.

Procedure

Send a GET request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}`.

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
{{collectionId}}	The ID of the collection

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections/
{{collectionId}} \
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes details of your collection.

1.4.3.1.1.1.2.3.1.1.3 Get Collections

This endpoint allows you to retrieve a list of all collections associated with a specific resource group.

Procedure

Send a GET request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections`

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections \
  --header 'AI-Resource-Group: {{resource_group}}'
```

```
  --header 'Authorization: Bearer {{access_token}}'
```

Pagination Support

You can use OData-compliant query parameters to manage large result sets:

Parameter	Type	Description
$top	Integer	Limits the number of results returned. for example., $top=10 returns 10 results.
$skip	Integer	Skips a specific number of results. for example, $skip=20 skips the first 20.
$count	Boolean	If true, includes the total count in the response. If false, includes the total number of records in the response.

For example:

```
curl --request GET
--url "$AI_API_URL/v2/lm/document-grounding/vector/collections?
$top=10&$skip=20&$count=true"
--header "AI-Resource-Group: {{resource_group}}"
--header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes a JSON array of collection objects.

1.4.3.1.1.1.2.3.1.1.4 Get a Collection

This endpoint allows you to retrieve details of a specific collection within a specified resource group.

Procedure

Send a GET request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections`
Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account

	Value
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
{{collectionId}}	The ID of the collection

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections/
{{collectionId}} \
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes details of your collection.

1.4.3.1.1.1.2.3.1.1.5 Get Collection Creation Status

This endpoint allows you to retrieve the status of a collection creation process using the collection ID.

Procedure

Send a GET request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections/{{id}}/creationStatus`.

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
{{id}}	The ID of the collection

Sample Code

```
curl --request GET \
```

```
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections/{{id}}/
creationStatus \
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes a the status of your collection creation.

Next Steps

After the collection has been created succesfully, you can add documents to your collection. For more information, see Create a Document [page 487].

1.4.3.1.1.1.2.3.1.1.6 Delete a Collection

This endpoint is used to delete a collection using its collection id.

Procedure

Send a DELETE request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}`

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
{{collectionId}}	The ID of the collection

Sample Code

```
curl --request DELETE \
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections/
{{collectionId}} \
```

```
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes a `location` header that contains the endpoint that can be used to get the status of deletion. Successful requests return a 202 response.

1.4.3.1.1.1.2.3.1.1.7 Get Collection Deletion Status

This endpoint allows you to retrieve the status of a collection deletion using the collection ID.

Procedure

Send a GET request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections/{{id}}/deletionStatus`

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
{{id}}	The ID of the collection

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections/{{id}}/
deletionStatus \
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes a the status of your collection deletion.

1.4.3.1.1.1.2.3.1.2 Documents

1.4.3.1.1.1.2.3.1.2.1 Create a Document

This endpoint allows you to add a document with associated metadata and chunks to a collection. Documents are units of text-based content that you upload into a collection, to be used for semantic search and grounding.

Prerequisites

You have created a document collection. For more information, see Vector API [page 477] and Create a Collection [page 480].

You know the ID of the collection that you want to add documents to. For more information, see Get a Collection [page 483].

Procedure

1. Send a POST request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents`.
   Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
Content-Type	application/json
{{collectionId}}	The ID of the collection
metadata <key> <value> pair	Your choice of metadata key value pair
chunks	Text content in chunks
metadata <key> <value> pair (chunks)	Your choice of metadata key value pair

{·} Sample Code

```
curl --request POST \
 --url  $AI_API_URL/v2/lm/document-grounding/vector/collections/
{{collectionId}}/documents\
 --header 'AI-Resource-Group: {{resource_group}}' \
 --header 'Content-Type: application/json' \
 --header 'Authorization: Bearer {{access_token}}'
  --data '{
```

```
  "documents": [
    {
      "metadata": [
        {
          "key": "url",
          "value": [
            "http://hello.com",
            "123"
          ]
        }
      ],
      "chunks": [
        {
          "content": "<chunk content 1>",
          "metadata": [  //
            {
              "key": "index",
              "value": [
                "1"
              ]
            }
          ]
        },
        {
          "content": "<chunk content 2>",
          "metadata": [
            {
              "key": "index",
              "value": [
                "2"
              ]
            }
          ]
        }
      ]
    }
  ]
}'
```

Results

The response is returned in JSON format, and includes a response code.

The response codes are:

- 201: Created (successful response)
- 400: The specification of the resource was incorrect
- 404: The specification of the resource was incorrect
- 422: There are validation issues with the data

Next Steps

To search for relevant chunks within a specific collection or across all collections based on a user query, use vector search. For more information, see Vector Search [page 498].

To search for relevant chunks across all collections based on a user query, use the Retrieval API. For more information, see Retrieval API [page 504].

To use your documents as part of an orchestration workflow, see Using the Grounding Module [page 510].

1.4.3.1.1.1.2.3.1.2.2 Update a Document

This endpoint allows you to update an existing document within a collection, including its associated metadata and chunks.

Procedure

Send a PATCH request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents`

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
Content-Type	application/json
{{collectionId}}	The ID of the collection
{{documentId}}	The ID of the document
metadata <key> <value> pair	Your choice of metadata key value pair
chunks	Text content in chunks
metadata <key> <value> pair (chunks)	Your choice of metadata key value pair

Sample Code

```
curl --request PATCH \
 --url $AI_API_URL/v2/lm/document-grounding/vector/collections/
{{collectionId}}/documents \
 --header 'AI-Resource-Group: {{resource_group}}' \
 --header 'Content-Type: application/json' \
 --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "documents": [
    {
      "id": {{documentId}},
      "metadata": [
        {
```

```
          "key": "url",
          "value": [
            "http://hello.com",
            "123"
          ]
        }
      ],
      "chunks": [
        {
          "content": "<This is an updated document chunk content>",
          "metadata": [
            {
              "key": "index",
              "value": [
                "1"
              ]
            }
          ]
        }
      ]
    }
  ]
}'
```

Results

The response is returned in JSON format, and includes a response code.

The reponse codes are:

- 200: OK (successful response)
- 400: The specification of the resource was incorrect
- 404: The specification of the resource was incorrect
- 422: There are validation issues with the data

1.4.3.1.1.1.2.3.1.2.3 Get All Documents in a Collection

This endpoint allows you to retrieve all documents associated with a specified collection ID.

Procedure

Send a GET request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents`

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
{{collectionId}}	The ID of the collection

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections/
{{collectionId}}/documents \
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
```

Pagination Support

You can use OData-compliant query parameters to manage large result sets:

Parameter	Type	Description
$top	Integer	Limits the number of results returned. for example., $top=10 returns 10 results.
$skip	Integer	Skips a specific number of results. for example, $skip=20 skips the first 20.
$count	Boolean	If true, includes the total count in the response. If false, includes the total number of records in the response.

For example:

```
curl --request GET
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/
documents?$top=10&$skip=20&$count=true"
  --header "AI-Resource-Group: {{resource_group}}"
  --header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes a JSON array of collection document objects.

1.4.3.1.1.1.2.3.1.2.4 Get A Document

This endpoint allows you to retrieve a specific document from a collection using the collection ID and document ID.

Procedure

Send a GET request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents/{{documentId}}`

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
{{collectionId}}	The ID of the collection
{{documentId}}	The ID of the document

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections/
{{collectionId}}/documents/{{documentId}} \
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes details of your document.

1.4.3.1.1.1.2.3.1.2.5 Delete a Document

This endpoint allows you to delete a specific document from a collection using the collection ID and document ID.

Procedure

Send a DELETE request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents/{{documentId}}`

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
{{collectionId}}	The ID of the collection
{{documentId}}	The ID of the document

Sample Code

```
curl --request DELETE \
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections/
{{collectionId}}/documents/{{documentId}} \
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes a response code.

The response codes are:

- 202: Deleted (successful response)
- 400: The specification of the resource was incorrect
- 404: The specification of the resource was incorrect
- 422: There are validation issues with the data

1.4.3.1.1.1.2.3.1.2.6 Delete Documents

This endpoint allows you to bulk delete specific documents using document IDs.

Procedure

Send a DELETE request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/documents`
Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<id1>..	The Ids of the documents that you want to delete.

Include the IDs of the documents that you want to delete in the `ids` field of the data.

Sample Code

```
curl --request DELETE \
  --url $AI_API_URL/v2/lm/document-grounding/vector/documents \
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
    "ids": [
      "<id1>",
      "<id2>",
      "<id3>"
    ]
}'
```

Results

The response is returned in JSON format, and includes a response code.

The response codes are:

- 202:
  - Deleted - document IDs successfully deleted
  - NotFound – document IDs not found and not deleted
- 400: The specification of the resource was incorrect
- 404: The specification of the resource was incorrect
- 422: There are validation issues with the data

1.4.3.1.1.1.2.3.2 Metadata

Context

Metadata is additional information, such as tags, categories, or custom attributes that you can add to your collections, documents, or chunks to help organize, filter, and search your content more effectively.

You can update metadata for multiple collections or multiple documents or multiple chunks in a single API request.

You can use the following operations:

Operation	Description
`add`	If the key is present, then it will add the new values. If the key is absent, it will add the key and the values.
`remove`	If the key is present, then it will remove the values from it.
`replace`	If the key is present, it will replace the entire list of values with the new values. If the key is absent, it will add the key and the values.
`delete_key`	If the key is present, then it will remove the key and all its values.

Limits

The following limits are the maximum number of updates that can be applied in each update:

- **5** update blocks per **PATCH request**. Ids can be grouped as required.
- **10** IDs per **update block**.
- **10** metadata updates per **update block**.
- For `remove` or `delete_key` operations: **1** operation per **update**.
- For `add` or `replace` operations: **2** operations per **update**.
- **10** values per **operation**.

Procedure

Send a PATCH request to the respective endpoint

Resource to be Updated	Endpoint
Collections	`$AI_API_URL/v2/lm/document-grounding/vector/collections/metadata`

Resource to be Updated	Endpoint
Documents	`$AI_API_URL/v2/lm/document-grounding/vector/documents/metadata`
Chunks	`$AI_API_URL/v2/lm/document-grounding/vector/chunks/metadata`

Populate the sample code with the following values:

Parameter	Description
`{{resource_group}}`	The AI resource group assigned to your account
`$AI_API_URL`	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
`{{access_token}}`	Your access token for SAP AI Core
`{{resource_id_1}}`	The ID of the collection, document, or chunk that you want to update. You can update multiple resources by providing a list of IDs.
`{{resource_id_2}}` (optional)	The ID of a second collection, document, or chunk that you'd like to update in the same request.

The sample provided completes the following tasks:

For `{{resource_id_1}}`:

- If the metadata key `"department"` is not present, the key will be added with values `"finance"` and `"legal"`.
- If the metadata key `"department"` is present, the values `"finance"` and `"legal"` will be added to the existing values.
- If the metadata key `"department"` is present, the value `"tech"` will be removed from the existing values.
- If the metadata key `"status"` exists, the key will be removed.

For `{{resource_id_2}}`:

- If the metadata key `"department"` is present, all of the values associated with the key will be replaced by the two values `"finance"` and `"legal"`.
- If the metadata key `"department"` is not present, the key will be added with values `"finance"` and `"legal"`

Sample Code

```
curl --request PATCH \
 --url $AI_API_URL/v2/lm/document-grounding/vector/collections/metadata \
 --header 'AI-Resource-Group: {{resource_group}}' \
 --header 'Content-Type: application/json' \
 --header 'Authorization: Bearer {{access_token}}'
  --data '{
    "updates": [
        {
            "ids": [
                {{resource_id_1}}
            ],
            "metadataUpdates": [
                {
```

```
                    "key": "department",
                    "operations": [
                        {
                            "op": "add",
                            "values": [
                                "finance",
                                "legal"
                            ]
                        },
                        {
                            "op": "remove",
                            "values": [
                                "tech"
                            ]
                        }
                    ]
                },
                {
                    "key": "status",
                    "operations": [
                        {
                            "op": "delete_key"
                        }
                    ]
                }
            ]
        },
        {
            "ids": [
                {{recource_id_2}}
            ],
            "metadataUpdates": [
                {
                    "key": "department",
                    "operations": [
                        {
                            "op": "replace",
                            "values": [
                                "finance",
                                "legal"
                            ]
                        }
                    ]
                }
            ]
        }
    ]
}'
```

Results

The response is returned in JSON format, and includes details of the updated resources and metadata.

1.4.3.1.1.1.2.3.3 Vector Search

This endpoint allows you to perform a search for relevant chunks within a specific collection or across all collections based on a user query. The response includes chunks that match the query, filtered by collection and document metadata.

Metadata Filtering Overview

Metadata Filtering lets you retrieve and filter documents, collections, and chunks with advanced metadata queries. It supports flexible, multi-level filtering to give you control over search results.

> ⓘ Note
>
> Filtering is available at the collection, document, and chunk levels.

The following are the match modes supports :

Key Match Modes

Simple Matching

- Provide `key-value` pairs in the `"chunkMetadata"`, `"documentMetadata"`, or `"collectionMetadata"` fields.
- Each entry defines a metadata key and a list of valid values.
- All specified `key-value` pairs must match for an item to be returned (logical **AND** across keys).
- Items containing all the given metadata values are included in the results.

Complex Matching

- Define `key-value` pairs using nested logical expressions in the filter field.
- Use logical operators (**AND**, **OR**) to combine multiple conditions.
- Each condition can define the metadata key, values, and the scope (collection, document, or chunk).
- Filters can be nested to create more advanced queries.

> → Tip
>
> Complex filtering is recommended because it offers greater flexibility and is the recommended approach.

Values Match Modes

Match Modes

- **ANY**
  - At least one metadata value matches.
  - This is the default match mode.
  - Defined only at injection time.
- **ALL**
  - All document metadata values must match exactly, or be a subset of the search metadata.
  - Supported only at the document level.

- Defined only at injection time.

Select Modes

- Supports `ignoreIfKeyAbsent` to control how missing keys are handled. This option can be set at query time and is supported only at the document level.
- The default is `False`.
- When set to `True`, documents that do not contain the specified key are also included in the results.

Procedure

Send a POST request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/search`

Populate the sample code with the following values:

Parameter	Description
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
query	Your user query
collectionIds	Enter `"*"` to search all collections or add a collectionId to search a specific collection. To search more than one specific collection, add `collectionIds` in a comma separated list.
documentMetadata	Filters based on metadata previously assigned

Example with Simple Matching

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/vector/search \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "query": "is Joule an AI Copilot?",
  "filters": [
    {
      "id": "String",
      "collectionIds": ["*"],
      "configuration": {},
      "collectionMetadata": [],
      "documentMetadata": [
        {
          "key": "url",
          "value": [
```

```
            "http://hello.com",
            "1234"
          ]
        }
      ],
      "chunkMetadata": []
    }
  ]
}'
```

Example with Complex Matching

Use the following examples to define grounding metadata filters. The example below shows a filter that retrieves content matching the following metadata specification:

- `key1=["value1"]` OR `key2=["value2"]`
- `(key1=["value1"]` AND `key2=["value2"])` OR `key3=["value3"]`
- `(key1=["value1"]` AND `key2=["value2"])` OR `key3=["value3"]`
- `key1=["value1"]` AND `key2=["value2"]`

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/vector/search \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "query": "is Joule an AI Copilot?",
  "filters": [
    {
      "id": "String",
      "collectionIds": ["*"],
      "configuration": {},
      "collectionMetadata": [],
      "documentMetadata": [],
      "chunkMetadata": [],
      "filter": {
        "operator": "and",
        "left": {
          "key": "key1",
          "value": [
          "value1"
          ],
          "scope": "document"
        },
        "right": {
          "key": "key2",
          "value": [
          "value2"
          ],
          "scope": "document"
        }
      }
    }
  ]
}'
```

Nested Complex Matching

The following example shows a complex filter with nested conditions. It retrieves all content where the chunk has `key1=value1`, and either the document has `key2=value2` or the collection has `key3=value3`.

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/vector/search \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "query": "is Joule an AI Copilot?",
  "filters": [
    {
      "id": "String",
      "collectionIds": ["*"],
      "configuration": {},
      "collectionMetadata": [],
      "documentMetadata": [],
      "chunkMetadata": [],
      "filter": {
        "operator": "and",
        "left": {
          "key": "key1",
          "value": [
            "value1"
          ],
          "scope": "chunk"
        },
        "right": {
          "operator": "or",
          "left": {
            "key": "key2",
            "value": [
              "value2"
            ],
            "scope": "document"
          },
          "right": {
            "key": "key3",
            "value": [
              "value3"
            ],
            "scope": "collection"
          }
        }
      }
    }
  ]
}'
```

Nested Complex Filter with Match Mode ALL

The document is created with metadata where `key1=["value11"]` uses the default match mode **ANY**, and `key2=["value21","value22"]` uses match mode **ALL**. The search request contains both keys at the document level.

Because the document's`key2` contains the full set of requested values, the document matches the filter and is returned.

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/vector/search \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "query": "is Joule an AI Copilot?",
  "filters": [
    {
      "id": "String",
      "collectionIds": ["*"],
      "configuration": {},
      "collectionMetadata": [],
      "documentMetadata": [],
      "chunkMetadata": [],
      "filter": {
        "operator": "and",
        "left": {
          "key": "key1",
          "value": [
          "value11"
          ],
          "scope": "document"
        },
        "right": {
          "key": "key2",
          "value": [
          “value21”, "value22", “value23”
          ],
          "scope": "document"
        }
      }
    }
  ]
```

The response is returned in JSON format, and includes relevant chunks based on similarity search. Each chunk object has a `searchScores` object, describing how relevant the chunk is to the search query.

The results also contain a `denseRetrievalScore` containing the cosine similarity score of the text embeddings.

Payload Attributes

Attribute	Type	Required	Description	Constraints
query	string	Yes	The search query text	maxLength: 2000, minLength: 1
filters	array	Yes	Array of search filters	N/A
filters[].id	string	Yes	Unique identifier for this search filter	N/A

Attribute	Type	Required	Description	Constraints
filters[].collectionIds	array of strings	Yes	List of collection IDs to search in. Use ["*"] to search all collections	N/A
filters[].configuration	object	Yes	Search configuration options	N/A
filters[].configuration.maxChunkCount	integer	No	Maximum number of chunks to return	> 0, cannot be used with maxDocumentCount
filters[].configuration.maxDocumentCount	integer	No	Maximum number of documents to return	> 0, cannot be used with maxChunkCount
filters[].collectionMetadata	array	No	Metadata filters for collections	maxItems: 2000
filters[].collectionMetadata[].key	string	Yes (if parent used)	Metadata key to filter on	maxLength: 1024
filters[].collectionMetadata[].value	array of strings	Yes (if parent used)	Acceptable values for the key	maxLength per item: 1024
filters[].documentMetadata	array	No	Metadata filters for documents	maxItems: 2000
filters[].documentMetadata[].key	string	Yes (if parent used)	Document metadata key to filter on	maxLength: 1024
filters[].documentMetadata[].value	array of strings	Yes (if parent used)	Acceptable values for the key	maxLength per item: 1024
filters[].documentMetadata[].matchMode	string	No	Matching mode for document metadata values	"ANY" or "ALL"
filters[].documentMetadata[].selectMode	array	No	Selection mode options	N/A
filters[].documentMetadata[].selectMode[]	string	No	Selection mode option	"ignoreIfKeyAbsent"
filters[].chunkMetadata	array	No	Metadata filters for chunks	maxItems: 2000
filters[].chunkMetadata[].key	string	Yes (if parent used)	Chunk metadata key to filter on	maxLength: 1024
filters[].chunkMetadata[].value	array of strings	Yes (if parent used)	Acceptable values for the key	maxLength per item: 1024

1.4.3.1.1.1.2.4 Retrieval API

The Retrieval API lets you retrieve repositories or collections created through the Vector API. It also lets you perform similarity searches on the vector database to obtain relevant chunks and documents.

Prerequisites

You have created a vector database. For more information, see Vector API [page 477].

Capacity Quotas and Input Limits

Retrieval Search – Query Limits

The following query limits apply:

Category	Description	Limit Type	Limit
Filters	Max filters per query	Input Limit	15
Collections	Max collection IDs per query	Input Limit	5,000
Results	Max chunk results per query	Input Limit	100
Results	Max document results per query	Input Limit	10

API Request Format

When you build your API requests, follow the format outlined in the *Retrieval* section of the Grounding API. For more information, see Grounding API.

1.4.3.1.1.1.2.4.1 Retrieve a List of All Data Repositories

This endpoint allows you to retrieve a list of all repositories or vector collections stored in the vector database.

Procedure

You can retrieve the list by sending a GET request to the endpoint `$AI_API_URL/v2/lm/document-grounding/retrieval/dataRepositories`.

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/retrieval/dataRepositories \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes an array of data repository objects.

Pagination Support

You can use OData-compliant query parameters to manage large result sets:

Parameter	Type	Description
`$top`	Integer	Limits the number of results returned. For example, `$top=10` returns 10 results.
`$skip`	Integer	Skips a specific number of results. For example, `$skip=20` skips the first 20.
`$count`	Boolean	If `true`, includes the total count in the response. If `false`, includes the total number of records in the response.

Example

This code returns a JSON array of data repository objects.

```
curl --request GET
  --url "$AI_API_URL/v2/lm/document-grounding/retrieval/dataRepositories?
$top=10&$skip=20&$count=true"
  --header "AI-Resource-Group: {{resource_group}}"
  --header 'Authorization: Bearer {{access_token}}'
```

1.4.3.1.1.1.2.4.2 Retrieve a Specific Data Respository

Context

This endpoint allows you to retrieve a specific data repository or vector collection stored in the vector database.

Procedure

1. You can retrieve a data repository or vector collection by sending a GET request to the endpoint `$AI_API_URL/v2/lm/document-grounding/retrieval/dataRepositories/{{repositoryId}}`. Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/retrieval/dataRepositories/
{{repositoryId}} \
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
```

This endpoint returns a response in JSON format. The response includes the specific data repository or vendor collection.

1.4.3.1.1.1.2.4.3 Retrieval Search

Context

This endpoint allows you to perform a similarity search between the user query and data repositories, returning relevant chunks that match the query.

Procedure

1. You can conduct this search by sending a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/retrieval/search`.
   Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<user query>	The query that you want to be answered from the vector database
dataRepositories	`'*'` searches through all collections. Alternatively, enter a collection ID to retrieve a chunk from that collection.
dataRepositoryType	Supported repository types are `vector` and `help.sap.com`. Data from `help.sap.com` is available for retrieval from the database by default.

Examples

Example with SAP HANA Vector Store, using all “*” data repositories

Sample Code

```
curl --request POST \
--url $AI_API_URL/v2/lm/document-grounding/retrieval/search \
--header 'AI-Resource-Group: {{resource_group}}' \
--header 'Authorization: Bearer {{access_token}}'
--data '{
  "query": "<user query>",
  "filters": [
    {
      "id": "string",
      "searchConfiguration": {
        "maxChunkCount": 1
      },
      "dataRepositories": ['*']
      ],
      "dataRepositoryType": "vector",
      "dataRepositoryMetadata": [
      {
          "key": "type",
          "value": [
            "custom"
          ]
        }
      ],
      "documentMetadata": [
        {
```

```
          "key": "url",
          "value": [
            "http://hello.com",
            "123"
          ]
        }
      ],
      "chunkMetadata": [
        {
          "key": "index",
          "value": [
            "1"
          ]
        }
      ]
    }
  ]
}'
```

Example with help.sap.com

Sample Code

```
curl --request POST \
--url $AI_API_URL/v2/lm/document-grounding/retrieval/search \
--header 'AI-Resource-Group: {{resource_group}}' \
--header 'Authorization: Bearer {{access_token}}'
--data '{
    "query": "<user-query>",
    "filters": [
        {
        "id": "string",
        "searchConfiguration": {},
        "dataRepositories": ["*"],
        "dataRepositoryType": "help.sap.com",
        "dataRepositoryMetadata": [],
        "documentMetadata": []
    }'
}'
```

Results

This endpoint returns a response in JSON format. The response includes relevant chunk identified using the similarity search.

Chunk objects may also contain the following scores:

- `searchScores`: the relevance of each chunk to the search query, reported from the Vector Search. For more information see, Vector Search [page 498].
- `postProcessingScore`: the score after post processing the original score, for example with rerankers, Not all post processing operations will have a `postProcessingScore`.

Post Processing

Results from multiple repositories can be merged, by passing the results of the `filters` to the `postProcessing` field.

You define the `"inputs":` to specify which filter results will be included in post processing

> ⚠ Restriction
>
> Only Cohere 3.5 Reranker is supported.

Merging Strategies

The following strategies are supported for combining results from different filters:

- `scoreReuse`: Merges the scores of multiple filter-results by sorting their scores.
- `reranker`: Reorders the results of mulitple filters by using a reranker and their content.
  To include metadata in the reranking, include your metadata in the `"boosting"` field. Metadata is not used as a filter, but enhances the relevance of a chunk where a query is related to the metadata.

Example

The following example uses the reranker strategy and retrieval results from a vector similarity search.

> {·} Sample Code

```
{
  "query": "How much leave do I have?",
  "filters": [
    {
      "id_": "filter_1",
      "searchConfiguration": {
        "maxChunkCount": 20,
        "maxDocumentCount": null
      },
      "dataRepositories": [
        "*"
      ],
      "dataRepositoryType": "vector",
      "dataRepositoryMetadata": [
        {
          "key": "lob",
          "value": [
            "sfsf"
          ]
        }
      ],
      "documentMetadata": [
        {
          "key": "country",
          "value": [
            "US"
          ],
```

```
          "selectMode": null
        }
      ],
      "chunkMetadata": []
    }
  ],
  "postProcessing": [
    {
      "id_": "27b9aadb-7c9e-4f1a-86b5-ea419d2d4e67",
      "maxChunkCount": 5,
      "strategy": {
        "type_": "reranker",
        "model": "cohere-3.5",
        "boosting": [
          {
            "key": "country",
            "value": "US",
            "scope": "filter_1"
          }
        ],
        "includeMetaData": false
      },
      "inputs": [
        {
          "id_": "filter_1"
        }
      ]
    }
  ]
}
```

1.4.3.1.1.2 Using the Grounding Module

Prerequisites

You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].

Procedure

Populate the sample code with the following values:

Field	Value
$RESOURCE_GROUP	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.

Field	Value
Bearer $TOKEN	Your access token for SAP AI Core
grounding_module_config.type	`"document_grounding_service"`
grounding_module_config.config.filters[].id	`"filter"`
grounding_module_config.config.filters[].data_repositories	Array specifying which repositories to use for document grounding or `"*"` to use all available data repositories
grounding_module_config.config.filters[].search_config.max_chunk_count	Integer that limits the maximum number of chunks to retrieve.
grounding_module_config.config.filters[].data_repository_type	`"vector"`
grounding_module_config.config.input_params	A reference to the variable containing your grounding query
grounding_module_config.config.output_param	A variable that stores your grounding output
llm_module_config	Your model configuration
templating_module_config.template	Your prompt template
input_params.groundingRequest	A string containing your grounding query

Examples

Example with SAP HANA Vector Store, using all “*” data repositories

Sample Code

```
curl --request POST "$ORCH_DEPLOYMENT_URL/completion" \
--header "content-type: application/json" \
--header "Authorization: Bearer $TOKEN" \
--header "ai-resource-group: $RESOURCE_GROUP" \
--data-raw '{
    "orchestration_config": {
        "module_configurations": {
            "grounding_module_config": {
                "type": "document_grounding_service",
                "config": {
                    "filters": [
                        {
                            "id": "filter",
                            "data_repositories": [
                                "*"
                            ],
                            "search_config": {
                                "max_chunk_count": 10
                            },
                            "data_repository_type": "vector"
                        }
                    ],
                    "input_params": [
                        "groundingRequest"
                    ],
                    "output_param": "groundingOutput"
                }
            },
            "llm_module_config": {
```

```
                "model_name": "<modelName>",
                "model_params": {},
                "model_version": "<modelVersion>"
            },
            "templating_module_config": {
                "template": [
                    {
                        "role": "user",
                        "content": "{{prompt}}"
                    }
                ],
                "defaults": {}
            }
        }
    },
    "input_params": {
        "groundingRequest": "{{grounding query}}"
    }
}'
```

Related Information

Leveraging Orchestration Capabilities to Enhance Responses
Libraries and SDKs

1.4.3.1.1.2.1 Contextualized Retrieval Using Metadata and Vector Search

You can choose to include metadata in the output structure of the grounding module in orchestration. You can leverage the metadata in your prompt template.

Prerequisites

You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].

You have onboarded a repository. For more information, see Generic Secrets for Grounding [page 398].

Context

Meta data inclusion is optional, and can be included using the `metadata_params` field in the completions endpoint of the API. The `metadata_params` field takes a list of comma separated strings.

Procedure

1. **Optional:** Retrieve supported metadata keys by sending your query to the endpoint `{{AI_API_Url}}/v2/lm/document-grounding/retrieval/search`.

   Sample Code

   ```
   curl --request POST \
     --url {{AI_API_URL}}/v2/lm/document-grounding/retrieval/search \
     --header 'AI-Resource-Group: <resource_group>' \
     --header 'Authorization: Bearer <access_token>' \
     --header 'Content-Type: application/json' \
     --data '{
       "query": "what is SAP Joule",
       "filters": [
           {
               "id": "string",
               "searchConfiguration": {
                   "maxChunkCount": 5
               },
               "dataRepositories": [
                   "*"
               ],
               "dataRepositoryType": "vector",
               "dataRepositoryMetadata": [
                   {
                       "key": "type",
                       "value": [
                           "custom"
                       ]
                   }
               ],
               "documentMetadata": [],
               "chunkMetadata": []
           }
       ]
   }'
   ```

   The response includes details of metadata parameters on the following levels: `dataRepository`, `document` and `chunk`.

2. To include metadata as part of your request, use `metadata_params` and include them as comma separated strings.

   The following example requests that `source` and `webUrl` metadata is included in the results

   Sample Code

   ```
   curl --request POST "$ORCH_DEPLOYMENT_URL/completion" \
   --header "content-type: application/json" \
   --header "Authorization: Bearer $TOKEN" \
   --header "AI-Resource-Group: $RESOURCE_GROUP" \
    --data '{
       "orchestration_config": {
           "module_configurations": {
               "grounding_module_config": {
                   "type": "document_grounding_service",
                   "config": {
                       "filters": [
                           {
                               "id": "filter1",
                               "data_repositories": [
                                   "*"
   ```

```
                            ],
                            "search_config": {},
                            "data_repository_type": "vector"
                        }
                    ],
                    "input_params": [
                        "groundingRequest"
                    ],
                    "output_param": "groundingOutput",
                    "metadata_params": [
                        "source",
                        "webUrl"
                    ]
                }
            },
            "llm_module_config": {
                "model_name": "gemini-1.5-pro",
                "model_params": {},
                "model_version": "001"
            },
            "templating_module_config": {
                "template": [
                    {
                        "role": "user",
                        "content": "You are a helpful assistant for
any queries for SAP Teched 2024.\nAnswer the grounding request by
providing relevant answers that fit to the request. \n\nRequest: {{ ?
groundingRequest }}\n\nReports:{{ ?groundingOutput }}"
                    }
                ],
                "defaults": {}
            },
            "filtering_module_config": {
                "input": {
                    "filters": [
                        {
                            "type": "azure_content_safety",
                            "config": {
                                "Hate": 2,
                                "SelfHarm": 2,
                                "Sexual": 2,
                                "Violence": 2
                            }
                        }
                    ]
                },
                "output": {
                    "filters": [
                        {
                            "type": "azure_content_safety",
                            "config": {
                                "Hate": 2,
                                "SelfHarm": 2,
                                "Sexual": 2,
                                "Violence": 2
                            }
                        }
                    ]
                }
            }
        }
    },
    "input_params": {
        "groundingRequest": "what is SAP Joule?"
    }
}'
```

Output Code

```
[
    [
        {
            "content": "Joule is the AI copilot that truly understands
your business. Joule revolutionizes how you interact with your SAP
business systems, making every touchpoint count and every task simpler.",
            "metadata": {
                "source": [
                    "/tmp/document_1.pdf"
                ],
                "webUrl": [
                    "https://<sharepoint_url>/sites/<Documents>/
sap_joule.pdf"
                ],
                "document_webUrl": [
                    "https://<sharepoint_url>/sites/<Documents>/
sap_joule.pdf"
                ]
            }
        },
        {
            "content": "AI-powered capabilities in Joule with custom-
developed enterprise skills tailored to your company's specific needs.
By integrating with both SAP and third-party applications, Joule can
instantly apply its newfound knowledge within SAP systems.",
            "metadata": {
                "source": [
                    "/tmp/document_1.pdf"
                ],
                "webUrl": [
                    "https://<sharepoint_url>/sites/<Documents>/
sap_joule.pdf"
                ],
                "document_webUrl": [
                    "https://<sharepoint_url>/sites/<Documents>/
sap_joule.pdf"
                ]
            }
        }
    ]
]
```

3. Leverage the metadata in your prompt

   Include instructions in your prompt for how the model should use the metadata.

Sample Code

```
"prompt_templating": {
  "messages_history": [
      {
      "role": "user",
      "content": "You are a helpful assistant for any queries for
SAP Teched 2024.\\nAnswer the grounding request by providing relevant
answers that fit to the request. \\n\\nRequest: {{ ?groundingRequest }}\\n\
\nReports:{{ ?groundingOutput }} Alongwith the original response, provide
another response by making use of metadata received in the grounding
output to provide more optimal response. Explain how the metadata was
used."
   }
  ],
   "defaults": {}
```

```
}
```

> **Note**
>
> If a metadata name is present on multiple levels, the result will include a `parent_KeyName` naming convention. The following example shows that `webUrl` is present on the chunk and document levels:
>
> **Output Code**
>
> ```
> "metadata": {
>   "webUrl": "<value from chunk metadata>",
>   "document_webUrl": "<value from document metadata>"
> }
> ```

1.4.3.1.2 Templating

The templating module is mandatory. It enables you to compose prompts and define placeholders. It then generates the final query that is sent to the model configuration module. Any placeholders that you define in your prompt can be filled at runtime. For example, the following template places the input text into the `{{ ?input }}` placeholder:

```
"templating_module_config": {
  "template": [
    {
      "role": "user",
      "content": "{{ ?input }}"
    }
  ]
}
```

The `{{ ?input }}` placeholder is filled using the contents of the `input_params.input` parameter:

```
"input_params": {
  "input": "A sample prompt to be sent to the model"
}
```

You can also set default values for the placeholders. For example, to request either 5 paraphrases or a user-specific number of paraphrases for a given phrase, you can use the following configuration:

```
"templating_module_config": {
  "template": [
    {
      "role": "user",
      "content": "Create {{ ?number }} paraphrases of {{ ?phrase }}"
    }
  ],
  "defaults": {
    "number": 5
  }
},
//... other configuration
"input_params": {
  "number": "3",
  "phrase": "Please respond as soon as possible, thanks."
```

```
}
```

Templates can also be retrieved from the prompt registry. For more information, see Prompt Registry [page 732].

Templates can be retrieved by ID (immutable) or the combination of scenario, name, and version (mutable). For examples:

Sample Code

```
"templating_module_config":{
  "template_ref":{
    "id": "d22342e7-1f91-4c52-a794-04833bc2574a"
    }
},
```

or

```
"templating_module_config":{
  "template_ref":{
    "scenario": "<scenario>",
    "name": "<name>",
    "version": "semver version e.g. 0.0.1 or `latest`"
  }
},
```

Input Image Support

Images can be provided as additional input where supported.

For example:

Sample Code

```
 "template": [
    {"role": "system", "content": "You are a helpful assistant."},
    {
        "role": "user",
        "content": [
            {"type": "text", "text": "what is in this image?"},
            {"type": "image_url", "image_url": {"url": "..."}},
            ],
        },

],
```

The URL for the image supports Base64 or a web-based URL of the image. Check with the model provider that your preferred upload method is supported.

```
"image_url": {
    "url":  f"{base64_image}"
    }
```

or

```
 "image_url": {
```

```
    "url": "https://some.jpg"

    }
```

1.4.3.1.2.1 Few-Shot Learning

Prerequisites

You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].

Process

The following example shows how you can configure the templating module to use a few-shot learning prompt.

```
curl --request POST $ORCH_DEPLOYMENT_URL/completion \
    --header 'content-type: application/json' \
    --header "Authorization: Bearer $TOKEN" \
    --header "AI-Resource-Group: $RESOURCE_GROUP" \
    --data-raw '{
    "orchestration_config": {
        "module_configurations": {
            "templating_module_config": {
                "template": [
                  {
                    "role": "system",
                    "content": "You classify input text into the two following
categories: Business, Economics, Tech, and other"
                  },
                  {
                    "role": "user",
                    "content": "input text: `Comcast launches prepaid plans`"
                  },
                  {
                    "role": "assistant",
                    "content": "Business"
                  },
                  {
                    "role": "user",
                    "content": "input text: `Slower Fed Pivot Weakens Rate-Cut
Bets Across Emerging Asia`"
                  },
                  {
                    "role": "assistant",
                    "content": "Economics"
                  },
                  {
                    "role": "user",
                    "content": "input text: {{?input}}"
                  }
              ]
```

```
            },
            "llm_module_config": {
                "model_name": "<ModelName>",
                "model_params": {
                    "max_tokens": 50,
                    "temperature": 0.1,
                    "frequency_penalty": 0.0,
                    "presence_penalty": 0.0
                }
            }
        }
    },
    "input_params": {
        "input": "Scaling up neural models has yielded significant advancements
in a wide array of tasks, particularly in language generation."
    }
}'
```

In this case, the template contains an array of messages, including a system message, as well as several user and assistant messages. The actual input to categories is configured with the final user message and the content `input text: {{?input}}`. The input parameter input is set to `Scaling up neural models has yielded significant advancements in a wide array of tasks, particularly in language generation.`.

Related Information

Leveraging Orchestration Capabilities to Enhance Responses
Libraries and SDKs

1.4.3.1.2.2 Chat

Orchestration can also be used in chat scenarios. The following example shows how to configure the templating module to use a chat prompt.

Prerequisites

You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].

Process

```
curl --request POST $ORCH_DEPLOYMENT_URL/completion \
    --header 'content-type: application/json' \
    --header "Authorization: Bearer $TOKEN" \
```

```
    --header "AI-Resource-Group: $RESOURCE_GROUP" \
    --data-raw '{
    "orchestration_config": {
      "module_configurations": {
        "templating_module_config": {
          "template": [
            {
              "role": "user",
              "content": "And now two more that sound like Goethe."
            }
          ]
        },
        "llm_module_config": {
          "model_name": "<ModelName>",
          "model_params": {
            "max_tokens": 300,
            "temperature": 0.1,
            "frequency_penalty": 0,
            "presence_penalty": 0
          }
        }
      }
    },
    "input_params": {
    },
    "messages_history": [
      {
        "role":"user",
        "content":"Create 3 paraphrases of I love you."
      },
      {
        "role":"assistant",
        "content":"1. I have deep affection for you.\n2. My feelings for you are
strong and caring.\n3. You mean everything to me."
      },
      {
        "role":"user",
        "content":"Can you please make them more romantic."
      },
      {
        "role":"assistant",
        "content":"1. My heart is filled with an intense passion for you.\n2.
You are the light of my life, and my love for you knows no bounds.\n3. I cherish
you more than words can express, my dear."
      },
      {
        "role":"user",
        "content":"Try sounding like shakespeare."
      },
      {
        "role":"assistant",
        "content":"1. Thou art the sun in my sky, and my love for thee doth
burn with unyielding fervor.\n2. Within mine heart, thou hast ignited a flame of
ardor that doth shine brightly for thee, my dearest.\n3. My love for thee doth
surpass all measure, for thou art the true essence of my being."
      }
    ]
  }'
```

As shown above, messages from the chat can be passed to the current request using the `messages_history` parameter. In this case, the chat history contains several user messages and assistant responses. The templating module is then configured with the current user message, which is used to generate the next assistant response. The templating module appends the current user message to the message history to generate the prompt that is sent to the LLM module.

The response contains the messages from the chat history and the response to the new message. The output of `module_results.templating` and the `orchestration_result.choices` results can be used as the message history for a subsequent inference request:


```
{
  "request_id": "5445f8d8-8b68-43c3-a149-26c1e6a88a22",
  "module_results": {
    "templating": [
      {
        "role": "user",
        "content": "Create 3 paraphrases of I love you."
      },
      {
        "role": "assistant",
        "content": "1. I have deep affection for you.\n2. My feelings for you
are strong and caring.\n3. You mean everything to me."
      },
      {
        "role": "user",
        "content": "Can you please make them more romantic."
      },
      {
        "role": "assistant",
        "content": "1. My heart is filled with an intense passion for you.\n2.
You are the light of my life, and my love for you knows no bounds.\n3. I cherish
you more than words can express, my dear."
      },
      {
        "role": "user",
        "content": "Try sounding like shakespeare."
      },
      {
        "role": "assistant",
        "content": "1. Thou art the sun in my sky, and my love for thee doth
burn with unyielding fervor.\n2. Within mine heart, thou hast ignited a flame of
ardor that doth shine brightly for thee, my dearest.\n3. My love for thee doth
surpass all measure, for thou art the true essence of my being."
      },
      {
        "role": "user",
        "content": "And now two more that sound like Goethe."
      }
    ],
    "llm": {
      "id": "chatcmpl-9kXqisJKnuNv1B4eXTUzqZEJSmzdC",
      "object": "chat.completion",
      "created": 1720880232,
      "model": "<ModelName>",
      "choices": [
        {
          "index": 0,
          "message": {
            "role": "assistant",
            "content": "1. In thy presence, my soul finds solace, for thou
art the embodiment of love's sweetest melody.\n2. Like a gentle breeze upon a
summer's eve, thy love doth caress my heart and fill it with eternal longing."
          },
          "finish_reason": "stop"
        }
      ],
      "usage": {
        "completion_tokens": 51,
        "prompt_tokens": 212,
        "total_tokens": 263
      }
    }
  },
```

```
  "orchestration_result": {
    "id": "chatcmpl-9kXqisJKnuNv1B4eXTUzqZEJSmzdC",
    "object": "chat.completion",
    "created": 1720880232,
    "model": "<ModelName>",
    "choices": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": "1. In thy presence, my soul finds solace, for thou art the
embodiment of love's sweetest melody.\n2. Like a gentle breeze upon a summer's
eve, thy love doth caress my heart and fill it with eternal longing."
        },
        "finish_reason": "stop"
      }
    ],
    "usage": {
      "completion_tokens": 51,
      "prompt_tokens": 212,
      "total_tokens": 263
    }
  }
}
```

Related Information

Leveraging Orchestration Capabilities to Enhance Responses
Libraries and SDKs

1.4.3.1.2.3 Minimal Call

Prerequisites

You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].

Process

A minimal call to orchestration contains only configurations of the required templating and model configuration modules. The curl command below shows how to make such a request.

```
curl --request POST $ORCH_DEPLOYMENT_URL/completion \
    --header 'content-type: application/json' \
    --header "Authorization: Bearer $TOKEN" \
```

```
    --header "AI-Resource-Group: $RESOURCE_GROUP" \
    --data-raw '{
  "orchestration_config": {
    "module_configurations": {
      "templating_module_config": {
        "template": [
          {
            "role": "user",
            "content": "Reply with `{{?text}}` in {{?language}}"
          }
        ],
        "defaults": {
          "language": "English"
        }
      },
      "llm_module_config": {
        "model_name": "gpt-4o-mini",
        "model_params": {
          "max_tokens": 50,
          "temperature": 0.1,
          "frequency_penalty": 0,
          "presence_penalty": 0
        },
        "model_version": "latest"
      }
    }
  },
  "input_params": {
    "text": "Orchestration is Working!",
    "language": "German"
  }
}'
```

This request configures the templating module with a single user message with two parameters: `text` and `language`. The `language` parameter is also configured with English as the default. The LLM module is configured to use gpt-4o-mini in the latest available version and a set of model parameters. The `input_params` field contains the values for the parameters `text` and `language`. These values are used during this request in the prompt sent to the model.

The response contains a `request_id`, the module results from each module that was executed, and the `orchestration_result`, which includes the response of the call to the model.

Output Code

```
{
  "request_id": "53fc2dcd-399d-4a2b-8bde-912b9f001fed",
  "module_results": {
    "templating": [
      {
        "role": "user",
        "content": "Reply with `Orchestration is Working!` in German"
      }
    ],
    "llm": {
      "id": "chatcmpl-9k8M3djXphXPWh2QkQm1YVtXK4Eki",
      "object": "chat.completion",
      "created": 1720782231,
      "model": "<ModelName>",
      "choices": [
        {
          "index": 0,
          "message": {
            "role": "assistant",
            "content": "Orchestrierungsdienst funktioniert!"
          },
```

```
          "finish_reason": "stop"
        }
      ],
      "usage": {
        "completion_tokens": 10,
        "prompt_tokens": 20,
        "total_tokens": 30
      }
    }
  },
  "orchestration_result": {
    "id": "chatcmpl-9k8M3djXphXPWh2QkQm1YVtXK4Eki",
    "object": "chat.completion",
    "created": 1720782231,
    "model": "gpt-4o-mini",
    "choices": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": "Orchestrierungsdienst funktioniert!"
        },
        "finish_reason": "stop"
      }
    ],
    "usage": {
      "completion_tokens": 10,
      "prompt_tokens": 20,
      "total_tokens": 30
    }
  }
}
```


The templating module result contains the user message with the filled in parameters. The LLM module result contains the response of the model execution. In this example, the LLM module result and the orchestration result are the same. However, they might differ, such as when the output filtering module filters the response.

Related Information

Leveraging Orchestration Capabilities to Enhance Responses
Libraries and SDKs

1.4.3.1.2.4 Tool Calling

You can incorporate specialist tools into your orchestration workflow for example to make an API call for current data or to execute custom code. LLMs decide that a tool call is needed based on your system prompt, messages and tool definition. You then execute your function code, return the results and the model will incorporate the results into its response.

You can define multiple tools, and use the decision making capabilities of generative AI models to guide you in their use.

→ Tip

For use cases including both prompt optimization and tool calling, the `FC:ExactMatch` metric is recommended. Other metrics may return results that are unreliable.

Example

The following code shows a tool containing a custom function for an API call to retrieve item quantity information from an inventory:

{·} Sample Code

```
import requests
def get_inventory_quantity(product_id):
    response = requests.get(f"https://my-store-api.com/v1/inventory?
product_id={product_id}")
    data = response.json()
    return data['product_id']['quantity']
```

You can include tools in your template by providing and outline of the function and its arguments in your template. You can include guidance on when a tool should be used in your system messages.

The following call includes an outline of the get_inventory_quantity function above in the `tools` parameter, with a description of its use case in the system role content. The `input_params` include a request for product ID 123456.

{·} Sample Code

```
curl --request POST \
  --url {{orchestration_deployment_url}}/completion \
  --header 'AI-Resource-Group: default' \
  --header 'Authorization: Bearer {{token}}' \
  --header 'content-type: application/json' \
  --data '{
  "orchestration_config": {
    "module_configurations": {
      "templating_module_config": {
        "template": [
          {
            "role": "system",
            "content": "You are a helpful assistant. You help
users to analyze inventory of the user'\''s store by calling the
get_inventory_quantity function."
          },
          {
            "role": "user",
            "content": "How many units of the product with ID {{ ?
product_id }} are in stock?"
          }
        ],
        "tools": [
          {
            "type": "function",
            "function": {
              "name": "get_inventory_quantity",
              "description": "Check available quantity for a product ID.",
              "strict": true,
```

```
              "parameters": {
                "type": "object",
                "properties": {
                  "product_id": {
                    "type": "integer"
                  }
                },
                "required": [
                  "product_id"
                ],
                "additionalProperties": false
              }
            }
          }
        ]
      },
      "llm_module_config": {
        "model_name": "gpt-4o",
        "model_params": {
          "max_tokens": 200
        }
      }
    }
  },
  "input_params": {
    "product_id": "123456"
  }
}'
```

The model uses the tool definition to decide that a function call is needed, and returns the function name and its input arguments in the format defined in the template. Use the information provided to call your custom function.

{·} Output Code

```
{
  "request_id": "6ebc7713-3ae5-48a8-a17b-53772e4024cc",
  "module_results": {
    "templating": [
      {
        "role": "system",
        "content": "You are a helpful assistant. You help users to analyze
inventory of the user's store by calling the get_inventory_quantity function."
      },
      {
        "content": "How many units of the product with ID 123456 are in
stock?",
        "role": "user"
      }
    ],
    "llm": {
      "id": "chatcmpl-BKP9iN0ustKi2OA7efsSi5yw31EO4",
      "object": "chat.completion",
      "created": 1744202958,
      "model": "gpt-4-turbo-2024-04-09",
      "system_fingerprint": "fp_5603ee5e2e",
      "choices": [
        {
          "index": 0,
          "message": {
            "role": "assistant",
            "content": "",
            "tool_calls": [
              {
                "id": "call_il3KDaSC5zm6naTOnYv5VSZT",
```

```
                "type": "function",
                "function": {
                  "name": "get_inventory_quantity",
                  "arguments": "{\"product_id\":123456}"
                }
              }
            ]
          },
          "finish_reason": "tool_calls"
        }
      ],
      "usage": {
        "completion_tokens": 17,
        "prompt_tokens": 89,
        "total_tokens": 106
      }
    }
  },
  "orchestration_result": {
    "id": "chatcmpl-BKP9iN0ustKi2OA7efsSi5yw31EO4",
    "object": "chat.completion",
    "created": 1744202958,
    "model": "gpt-4-turbo-2024-04-09",
    "system_fingerprint": "fp_5603ee5e2e",
    "choices": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": "",
          "tool_calls": [
            {
              "id": "call_il3KDaSC5zm6naTOnYv5VSZT",
              "type": "function",
              "function": {
                "name": "get_inventory_quantity",
                "arguments": "{\"product_id\":123456}"
              }
            }
          ]
        },
        "finish_reason": "tool_calls"
      }
    ],
    "usage": {
      "completion_tokens": 17,
      "prompt_tokens": 89,
      "total_tokens": 106
    }
  }
}
```


When you have called your custom function, use the response to extend your LLM call.

In the following example, the function call has returned a quantity of 25 items in integer format. This information is inserted into the `template` in the `tool` role.

Sample Code

```
curl --request POST \
  --url {{orchestration_deployment_url}}/completion \
  --header 'AI-Resource-Group: default' \
  --header 'Authorization: Bearer {{token}}' \
  --header 'content-type: application/json' \
  --data '{
  "orchestration_config": {
```

```
    "module_configurations": {
      "templating_module_config": {
        "template": [
          {
            "role": "system",
            "content": "You are a helpful assistant. You help
users to analyze inventory of the user'\''s store by calling the
get_inventory_quantity function."
          },
          {
            "role": "user",
            "content": "How many units of the product with ID {{ ?
product_id }} are in stock?"
          },
          # append the function call message
          {
            "role": "assistant",
            "content": "",
            "tool_calls": [
              {
                "id": "call_il3KDaSC5zm6naTOnYv5VSZT",
                "type": "function",
                "function": {
                  "name": "get_inventory_quantity",
                  "arguments": "{\"product_id\":123456}"
                }
              }
            ]
          },
          # append function result
          {
            "role": "tool",
            "tool_call_id": "call_il3KDaSC5zm6naTOnYv5VSZT",
            "content": "25"
          }
        ],
        "tools": [
          {
            "type": "function",
            "function": {
              "name": "get_inventory_quantity",
              "description": "Check available quantity for a product ID.",
              "strict": true,
              "parameters": {
                "type": "object",
                "properties": {
                  "product_id": {
                    "type": "integer"
                  }
                },
                "required": [
                  "product_id"
                ],
                "additionalProperties": false
              }
            }
          }
        ]
      },
      "llm_module_config": {
        "model_name": "gpt-4",
        "model_params": {
          "max_tokens": 200
        }
      }
    }
  },
  "input_params": {
```

```
    "product_id": "123456"
  }
}'
```

The model answers the query, incorporating the result of the function call into its response:

Output Code

```
{
  "request_id": "196d5acc-9140-4501-8acf-9de84c36801f",
  "module_results": {
    "templating": [
      {
        "role": "system",
        "content": "You are a helpful assistant. You help users to analyze
inventory of the user's store by calling the get_inventory_quantity function."
      },
      {
        "content": "How many units of the product with ID 123456 are in
stock?",
        "role": "user"
      },
      {
        "role": "assistant",
        "content": "",
        "tool_calls": [
          {
            "id": "call_il3KDaSC5zm6naTOnYv5VSZT",
            "type": "function",
            "function": {
              "name": "get_inventory_quantity",
              "arguments": "{\"product_id\":123456}"
            }
          }
        ]
      },
      {
        "role": "tool",
        "tool_call_id": "call_il3KDaSC5zm6naTOnYv5VSZT",
        "content": "25"
      }
    ],
    "llm": {
      "id": "chatcmpl-BKPKM10RalAlw2drape8GUfSe60sZ",
      "object": "chat.completion",
      "created": 1744203618,
      "model": "gpt-4-turbo-2024-04-09",
      "system_fingerprint": "fp_5603ee5e2e",
      "choices": [
        {
          "index": 0,
          "message": {
            "role": "assistant",
            "content": "There are 25 units of the product with ID 123456 in
stock."
          },
          "finish_reason": "stop"
        }
      ],
      "usage": {
        "completion_tokens": 17,
        "prompt_tokens": 113,
        "total_tokens": 130
      }
    }
  },
```

```
  "orchestration_result": {
    "id": "chatcmpl-BKPKM10RalAlw2drape8GUfSe60sZ",
    "object": "chat.completion",
    "created": 1744203618,
    "model": "gpt-4-turbo-2024-04-09",
    "system_fingerprint": "fp_5603ee5e2e",
    "choices": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": "There are 25 units of the product with ID 123456 in
stock."
        },
        "finish_reason": "stop"
      }
    ],
    "usage": {
      "completion_tokens": 17,
      "prompt_tokens": 113,
      "total_tokens": 130
    }
  }
}
```


1.4.3.1.3 Input Translation

The input translation module is optional. It allows you to translate LLM text prompts into a chosen target language.

The input translation module helps improve answer quality when the configured model performs better with input in a specific language, for example English.

To use input translation, configure it as part of your orchestration workflow.

The module supports SAP's Document Translation service (part of the SAP Translation Hub).

> ⚠ Caution
>
> Input translation is performed **before** any configured input masking. Any content added to the prompt as part of prompt templating is sent to the translation service unmodified.

1.4.3.1.3.1 Enhance Model Consumption with Input Translation

To configure input translation, add an `input_translation_module_config` section to the `module_configurations` of your `orchestration_config`.

For example:

Sample Code

```
{
"orchestration_config": {
    "module_configurations": {
        ...
        "input_translation_module_config": {
            "type": "sap_document_translation",
            "config": {
                "source_language": "de-DE",
                "target_language": "en-US"
            }
        }
        ...
    }
    }
    ...
}
```

The `input_translation_module_config` object must specify the keys::

- `type`: specifying the translation service being called.
  The following values are supported:
  - `sap_document_translation`
- `config`: containing the parameters that are sent to the service to configure translation.
  The following parameters are supported:
  - `source_language` (optional)
  - `target_language` (mandatory)

The source language is the language of the input text. If no source language is specified, the SAP Translation Service will try to detect the source language.

→ Remember

For texts containing multiple languages, the service might be unable to detect the source language.

The target language is the language that the input will be translated and is mandatory.

For more information on possible languages and translation pairs, see Supported Languages of the document translation service in SAP Translation Hub.

Example

The following example shows a summarizing support assistant and uses the translation module to translate a support request message from German to English before sending it to the LLM for summarization.

Sample Code

```
{
    "orchestration_config": {
        "module_configurations": {
            "templating_module_config": {
                "template": [
```

```
                {
                "role": "system",
                "content": "You are a helpful support assistant. Your task
is to summarize a given support request for the human support team. \n Your
proceed as follows: Summarize the issue for the human support team. Provide
your answer in the following format:\n - Sentiment: [your sentiment analysis]
\n - Key Theme: [theme of the support issue] \n - Contact: [any contact
information available in the issue]"
                },
                {
                "role": "user",
                "content": "Support Request: '''{{?support-request}}'''"
                }
                ]
            },
            "llm_module_config": {
                "model_name": "gpt-4o-mini"
                },
            "input_translation_module_config": {
                "type": "sap_document_translation",
                "config": { // parameters specific to SAP Document
Translation that are sent to the service
                    "source_language": "de-DE",
                    "target_language": "en-US"
                }
            }
        }
    },
    "input_params": {
        "support-request": "Subject: Bestellung #1234567890 Verspätet - John
Johnson Nachricht: Halle, ich schreibe ihnen um mich nach dem Status meiner
Bestellung mit der Bestellnr. +1234567890 zu erkundigen. Die Lieferung war
eigentlich für gestern geplant, ist bisher jedoch nicht erfolgt. Mein Name
ist John Johnson und meine Lieferadresse lautet 125 Cole Meadows Drive Palo
Alto, California 94301. Bitte lassen Sie mich per Telefon unter der Nummer +1
505802 2172 wissen, wann ich mit meiner Lieferung rechnen kann. Danke!"
    }
}
```

1.4.3.1.4 Data Masking

The data masking module is optional and serves to anonymize or pseudonymize personally identifiable information from the input for selected entities.

There are two ways in which you can mask data:

- Anonymization replaces personally identifiable information in chosen categories with a `MASKED_ENTITY` placeholder. Anonymized data can't be unmasked because the original data isn't retained.
- Pseudonymization substitutes personally identifiable information in selected categories with a `MASKED_ENTITY_ID` placeholder. Pseudonymized data can be unmasked in the response.

The module currently supports the following anonymization services:

- SAP Data Privacy Integration - Anonymization

SAP Data Privacy Integration - Anonymization

SAP Data Privacy Integration - Anonymization service recognizes the following entity categories:

Entity	Description	Scope
profile-person	Person Name	English
profile-org	Organization	SAP customers + Fortune 1000 companies
profile-university	University	Organization identified as a public University
profile-location	Location	United States
profile-phone	Phone Number	International phone numbers in the following format: +country/region code, any prefix and phone number
profile-address	U.S. Physical Addresses	United States
profile-email	E-mail addresses	E-mail addresses
profile-sapids-internal	SAP Staff User ID Numbers	User ID: Sequence of characters that starts with either C, I, or D followed by 6-8 digits.
profile-sapids-public	Public SAP Accounts	S-user ID: Sequence of characters that starts with S and followed by 6-11 digits. P-user ID: Sequence of characters that starts with P followed by 10 digits.
profile-url	URL	Only anonymize URLs that can be accessed by the user
profile-username-password	Username and Passwords	Detected if words similar to user, user name, pass, passwords, and so on.
profile-nationalid	National ID Number	20+ countries/regions (for example, E.U., South America)
profile-iban	International Bank Account Number (IBAN)	70+ countries/regions (E.U., Middle East, South America)
profile-ssn	Social Security Number (SSN)/Social Insurance Number (SIN)	United States and Canada
profile-credit-card-number	Credit Card Number	Global
profile-passport	Passport Number	30+ countries/regions (for example, E.U., Asia Pacific, North America)
profile-driverlicense	Driver's License	30+ countries/regions (for example, E.U., Asia Pacific)
profile-nationality	Nationality	190+ country/region names, official country/region names, and country/region codes
profile-religious-group	Religious Group	200+ religious groups
profile-political-group	Political Parties or Groups	100+ political parties or groups
profile-pronouns-gender	Gender pronoun	Global

Entity	Description	Scope
profile-gender	Gender	Global
profile-sexual-orientation	Sexual Orientation	Global
profile-trade-union	Trade Union	Global
profile-sensitive-data	Nationality, Religious_Group, Gender, Political_Group, pronoun gender, ethnicity, sexual orientation, Trade union	Mixed
profile-ethnicity	Ethnicity	Global

To mask entities based on regular expressions, use custom entities. For more information, see Configuring Replacement Methods [page 369].

> ⚠ Caution
>
> The masking service can obscure personally identifiable information in prompts. However, since it relies on automated detection mechanisms, it can't guarantee that all such information is identified and masked.
>
> Anonymization replaces personally identifiable information in an irreversible way. As a result, context can be lost, which can limit the model's ability to process the input. For instance, if tasked with writing a story about Michael and Donna, anonymization of profile-person results in a story about `MASKED_PERSON` and `MASKED_PERSON`, making it impossible to distinguish between the two.

1.4.3.1.4.1 Enhancing Model Consumption with Data Masking

Prerequisites

You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].

Process

In the following example, we use the data masking module to anonymize people, organizations, and contact information in the input. The allow list contains the entries “Harvard University” and “Boston”, and so these are not masked. Here, input is masked before the model is called. The data in the model's output cannot be unmasked.

> {·} Sample Code

```
curl --request POST $ORCH_DEPLOYMENT_URL/completion \
    --header 'content-type: application/json' \
```

```
    --header "Authorization: Bearer $TOKEN" \
    --header "AI-Resource-Group: $RESOURCE_GROUP" \
    --data-raw '{
  "orchestration_config": {
    "module_configurations": {
      "templating_module_config": {
        "template": [
          {
            "role": "user",
            "content": "Summarize the following CV in 10 sentences: {{?
orgCV}}"
          }
        ]
      },
      "llm_module_config": {
        "model_name": "gpt-4",
        "model_params": {
          "max_tokens": 300,
          "temperature": 0.1,
          "frequency_penalty": 0,
          "presence_penalty": 0
        }
      },
      "masking_module_config": {
        "masking_providers": [
          {
            "type": "sap_data_privacy_integration",
            "method": "anonymization",
            "entities": [
              {"type": "profile-email"},
              {"type": "profile-person"},
              {"type": "profile-phone"},
              {"type": "profile-org"},
              {"type": "profile-location"}
              ],
              "allowlist": ["Harvard University", "Boston"] // "Harvard
University" and "Boston" will be preserved and not masked
          }
        ]
      }
    }
  },
  "input_params": {
    "orgCV": "Patrick Morgan \n +49 (970) 333-3833 \n
patric.morgan@hotmail.com \n\n Highlights \n - Strategic and financial
planning expert \n - Accurate forecasting \n - Process implementation \n -
Staff leadership and development \n - Business performance improvement \n -
Proficient in SAP,  Excel VBA\n\n Education \n Master of Science: Finance -
2014 \n Harvard University, Boston \n\n Bachelor of Science: Finance - 2011
\n Harvard University, Boston \n\n\n Certifications \n Certified Management
Accountant \n\n\n Summary \n Skilled Financial Manager adept at increasing
work process efficiency and profitability through functional and technical
analysis. Successful at advising large corporations, small businesses, and
individual clients. Areas of expertise include asset allocation, investment
strategy, and risk management. \n\n\n Experience \n Finance Manager - 09/2016
to 05/2018 \n M&K Group, York \n - Manage the modelling, planning, and
execution of all financial processes. \n - Carry short and long-term custom
comprehensive financial strategies to reach company goals. \n - Recommended
innovative alternatives to generate revenue and reduce unnecessary costs.
\n - Employed advanced deal analysis, including hands-on negotiations with
potential investors. \n - Research market trends and surveys and use
information to stimulate business. \n\n Finance Manager - 09/2013 to 05/2016
\n Ago Group, Chicago \n - Drafted executive analysis reports highlighting
business issues, potential risks, and profit opportunities. \n - Recommended
innovative alternatives to generate revenue and reduce unnecessary costs.
\n - Employed advanced deal analysis, including hands-on negotiations with
potential investors. \n - Analysed market trends and surveys and used
information to revenue growth."
```

```
  }
}'
```

As shown in the response, the data masking module masks the configured entities before sending them to the model. The model then operates on this masked data and can still provide a summary.

⚠ Caution

The `module_results` field may contain sensitive information, so we recommend not displaying its contents to end users.

{·} Output Code


```
{
  "request_id": "0b32ef8d-5f92-43de-ae2b-ae23cecba662",
  "module_results": {
    "templating": [
      {
        "role": "user",
        "content": "Summarize the following CV in 10 sentences: Patrick
Morgan \n +49 (970) 333-3833 \n patric.morgan@hotmail.com \n\n Highlights
\n - Strategic and financial planning expert \n - Accurate forecasting
\n - Process implementation \n - Staff leadership and development \n -
Business performance improvement \n - Proficient in SAP,  Excel VBA\n\n
Education \n Master of Science: Finance - 2014 \n Harvard University, Boston
\n\n Bachelor of Science: Finance - 2011 \n Harvard University, Boston
\n\n\n Certifications \n Certified Management Accountant \n\n\n Summary
\n Skilled Financial Manager adept at increasing work process efficiency
and profitability through functional and technical analysis. Successful
at advising large corporations, small businesses, and individual clients.
Areas of expertise include asset allocation, investment strategy, and risk
management. \n\n\n Experience \n Finance Manager - 09/2016 to 05/2018 \n
M&K Group, York \n - Manage the modelling, planning, and execution of all
financial processes. \n - Carry short and long-term custom comprehensive
financial strategies to reach company goals. \n - Recommended innovative
alternatives to generate revenue and reduce unnecessary costs. \n - Employed
advanced deal analysis, including hands-on negotiations with potential
investors. \n - Research market trends and surveys and use information to
stimulate business. \n\n Finance Manager - 09/2013 to 05/2016 \n Ago Group,
Chicago \n - Drafted executive analysis reports highlighting business issues,
potential risks, and profit opportunities. \n - Recommended innovative
alternatives to generate revenue and reduce unnecessary costs. \n - Employed
advanced deal analysis, including hands-on negotiations with potential
investors. \n - Analysed market trends and surveys and used information to
revenue growth."
      }
    ],
    "input_masking": {
      "message": "Masking module executed successfully on LLM input.",
      "data": {
        "masked_template": [
          {
            "role": "user",
            "content": "Summarize the following CV in 10 sentences:
MASKED_PERSON \n MASKED_PHONE_NUMBER \n MASKED_EMAIL \n\n Highlights \n
- Strategic and financial planning expert \n - Accurate forecasting \n
- Process implementation \n - Staff leadership and development \n -
Business performance improvement \n - Proficient in SAP, Excel VBA\n\n
Education \n Master of Science: Finance - 2014 \n Harvard University, Boston
\n\n Bachelor of Science: Finance - 2011 \n Harvard University, Boston
\n\n\n Certifications \n Certified Management Accountant \n\n\n Summary
\n Skilled Financial Manager adept at increasing work process efficiency
and profitability through functional and technical analysis. Successful
at advising large corporations, small businesses, and individual clients.
```

```
Areas of expertise include asset allocation, investment strategy, and risk
management. \n\n\n Experience \n Finance Manager - 09/2016 to 05/2018
\n MASKED_ORG, MASKED_LOCATION \n - Manage the modelling, planning, and
execution of all financial processes. \n - Carry short and long-term custom
comprehensive financial strategies to reach company goals. \n - Recommended
innovative alternatives to generate revenue and reduce unnecessary costs.
\n - Employed advanced deal analysis, including hands-on negotiations with
potential investors. \n - Research market trends and surveys and use
information to stimulate business. \n\n Finance Manager - 09/2013 to 05/2016
\n MASKED_ORG, MASKED_LOCATION \n - Drafted executive analysis reports
highlighting business issues, potential risks, and profit opportunities.
\n - Recommended innovative alternatives to generate revenue and reduce
unnecessary costs. \n - Employed advanced deal analysis, including hands-
on negotiations with potential investors. \n - Analysed market trends and
surveys and used information to revenue growth."
          }
        ]
      }
    },
    "llm": {
      "id": "chatcmpl-A5vfibe3l7gpZLPKQnym4NLmknErY",
      "object": "chat.completion",
      "created": 1725976694,
      "model": "gpt-4",
      "choices": [
        {
          "index": 0,
          "message": {
            "role": "assistant",
            "content": "The CV belongs to a skilled Financial Manager with
expertise in strategic and financial planning, accurate forecasting, process
implementation, staff leadership and development, and business performance
improvement. They are proficient in SAP and Excel VBA. They hold a Master
of Science in Finance from an undisclosed institution and location, and
a Bachelor of Science in Finance from another undisclosed institution
and location. They are also a Certified Management Accountant. Their
experience includes managing financial processes, implementing financial
strategies, recommending cost-saving measures, and conducting deal analysis
and market research. They have worked as a Finance Manager for two different
organizations from 2013 to 2018."
          },
          "finish_reason": "stop"
        }
      ],
      "usage": {
        "completion_tokens": 124,
        "prompt_tokens": 354,
        "total_tokens": 478
      }
    }
  },
  "orchestration_result": {
    "id": "chatcmpl-A5vfibe3l7gpZLPKQnym4NLmknErY",
    "object": "chat.completion",
    "created": 1725976694,
    "model": "gpt-4",
    "choices": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": "The CV belongs to a skilled Financial Manager with
expertise in strategic and financial planning, accurate forecasting, process
implementation, staff leadership and development, and business performance
improvement. They are proficient in SAP and Excel VBA. They hold a Master
of Science in Finance from an undisclosed institution and location, and
a Bachelor of Science in Finance from another undisclosed institution
and location. They are also a Certified Management Accountant. Their
```

```
experience includes managing financial processes, implementing financial
strategies, recommending cost-saving measures, and conducting deal analysis
and market research. They have worked as a Finance Manager for two different
organizations from 2013 to 2018."
        },
        "finish_reason": "stop"
      }
    ],
    "usage": {
      "completion_tokens": 124,
      "prompt_tokens": 354,
      "total_tokens": 478
    }
  }
}
```


When you use `"method": "pseudonymization"` instead of `"method": "anonymization"` in the masking module configuration, the data is pseudonymized before it's sent to the model. Additionally, the system checks the model's response for any data that can be unmasked before sending out the final response.

Output Code


```
{
  "request_id": "f5d14279-9cbe-4f35-815d-ea2df5f6d9d4",
  "module_results": {
    "templating": [
      {
        "role": "user",
        "content": "Summarize the following CV in 10 sentences: Patrick
Morgan \n +49 (970) 333-3833 \n patric.morgan@hotmail.com \n\n Highlights
\n - Strategic and financial planning expert \n - Accurate forecasting
\n - Process implementation \n - Staff leadership and development \n -
Business performance improvement \n - Proficient in SAP,  Excel VBA\n\n
Education \n Master of Science: Finance - 2014 \n Harvard University, Boston
\n\n Bachelor of Science: Finance - 2011 \n Harvard University, Boston
\n\n\n Certifications \n Certified Management Accountant \n\n\n Summary
\n Skilled Financial Manager adept at increasing work process efficiency
and profitability through functional and technical analysis. Successful
at advising large corporations, small businesses, and individual clients.
Areas of expertise include asset allocation, investment strategy, and risk
management. \n\n\n Experience \n Finance Manager - 09/2016 to 05/2018 \n
M&K Group, York \n - Manage the modelling, planning, and execution of all
financial processes. \n - Carry short and long-term custom comprehensive
financial strategies to reach company goals. \n - Recommended innovative
alternatives to generate revenue and reduce unnecessary costs. \n - Employed
advanced deal analysis, including hands-on negotiations with potential
investors. \n - Research market trends and surveys and use information to
stimulate business. \n\n Finance Manager - 09/2013 to 05/2016 \n Ago Group,
Chicago \n - Drafted executive analysis reports highlighting business issues,
potential risks, and profit opportunities. \n - Recommended innovative
alternatives to generate revenue and reduce unnecessary costs. \n - Employed
advanced deal analysis, including hands-on negotiations with potential
investors. \n - Analysed market trends and surveys and used information to
revenue growth."
      }
    ],
    "input_masking": {
      "message": "Masking module executed successfully on LLM input.",
      "data": {
        "masked_template": [
          {
            "role": "user",
            "content": "Summarize then following CV in 10 sentences:
MASKED_PERSON_1 \n MASKED_PHONE_NUMBER_1 \n MASKED_EMAIL_1 \n\n Highlights
\n - Strategic and financial planning expert \n - Accurate forecasting
```

```
\n - Process implementation \n - Staff leadership and development \n -
Business performance improvement \n - Proficient in SAP, Excel VBA\n\n
Education \n Master of Science: Finance - 2014 \n Harvard University, Boston
\n\n Bachelor of Science: Finance - 2011 \n Harvard University, Boston
\n\n\n Certifications \n Certified Management Accountant \n\n\n Summary
\n Skilled Financial Manager adept at increasing work process efficiency
and profitability through functional and technical analysis. Successful
at advising large corporations, small businesses, and individual clients.
Areas of expertise include asset allocation, investment strategy, and risk
management. \n\n\n Experience \n Finance Manager - 09/2016 to 05/2018 \n
MASKED_ORG_1, MASKED_LOCATION_1 \n - Manage the modelling, planning, and
execution of all financial processes. \n - Carry short and long-term custom
comprehensive financial strategies to reach company goals. \n - Recommended
innovative alternatives to generate revenue and reduce unnecessary costs.
\n - Employed advanced deal analysis, including hands-on negotiations with
potential investors. \n - Research market trends and surveys and use
information to stimulate business. \n\n Finance Manager - 09/2013 to 05/2016
\n MASKED_ORG_2, MASKED_LOCATION_2 \n - Drafted executive analysis reports
highlighting business issues, potential risks, and profit opportunities.
\n - Recommended innovative alternatives to generate revenue and reduce
unnecessary costs. \n - Employed advanced deal analysis, including hands-
on negotiations with potential investors. \n - Analysed market trends and
surveys and used information to revenue growth."
          ]
        }
      },
      "llm": {
        "id": "chatcmpl-A5veOHbkabo04FugU7ysHeWeH7ZZC",
        "object": "chat.completion",
        "created": 1725976612,
        "model": "gpt-4",
        "choices": [
          {
            "index": 0,
            "message": {
              "role": "assistant",
              "content": "MASKED_PERSON_1 is a skilled Financial Manager with
expertise in strategic and financial planning, accurate forecasting, process
implementation, staff leadership and development, and business performance
improvement. They are proficient in SAP and Excel VBA. They hold a
Master of Science in Finance from Harvard University and are a Certified
Management Accountant. Their experience includes advising large corporations,
small businesses, and individual clients on asset allocation, investment
strategy, and risk management. They have worked as a Finance Manager
for MASKED_ORG_1 and MASKED_ORG_2, where they managed financial processes,
implemented financial strategies, recommended cost-saving measures, and
conducted advanced deal analysis. They also have experience in researching
market trends to stimulate business growth."
            },
            "finish_reason": "stop"
          }
        ],
        "usage": {
          "completion_tokens": 147,
          "prompt_tokens": 378,
          "total_tokens": 525
        }
      }
    },
    "orchestration_result": {
      "id": "chatcmpl-A5veOHbkabo04FugU7ysHeWeH7ZZC",
      "object": "chat.completion",
      "created": 1725976612,
      "model": "gpt-4",
      "choices": [
        {
          "index": 0,
          "message": {
```

```
          "role": "assistant",
          "content": "Patrick Morgan is a skilled Financial Manager with
expertise in strategic and financial planning, accurate forecasting, process
implementation, staff leadership and development, and business performance
improvement. They are proficient in SAP and Excel VBA. They hold a Master
of Science in Finance from Harvard University and are a Certified Management
Accountant. Their experience includes advising large corporations, small
businesses, and individual clients on asset allocation, investment strategy,
and risk management. They have worked as a Finance Manager for M&K Group
and Ago Group, where they managed financial processes, implemented financial
strategies, recommended cost-saving measures, and conducted advanced deal
analysis. They also have experience in researching market trends to stimulate
business growth."
        },
        "finish_reason": "stop"
      }
    ],
    "usage": {
      "completion_tokens": 147,
      "prompt_tokens": 378,
      "total_tokens": 525
    }
  }
}
```

When the grounding module is used, the masking module also supports masking of the grounding input before retrieval. By default, this feature is disabled. To enable it the parameter `mask_grounding_input` in the `masking_module_config` can be used as follows:

Sample Code

```
{
  "orchestration_config": {
    "module_configurations": {
      "masking_module_config": {
        "masking_providers": [
          {
            "type": "sap_data_privacy_integration",
            "method": "pseudonymization",
            "entities": [
              {"type": "profile-person"},
              {"type": "profile-email"}
            ],
            "mask_grounding_input": {"enabled": true}
          }
        ]
        ...
      }
      ...
    }
  }
}
```

Configuring Replacement Methods

You can configure the replacement method for each entity in the `masking_module_config`. The following methods are supported:

- `fabricated_data`: Replaces the entity with a randomly generated value of the same type. For example, a person's name might be replaced with another name, and a phone number with a different phone number.
- `constant`: Replaces the entity with a specified constant value.
  - In anonymization, the constant value is used directly without any modification (for example, `MASKED_PERSON`).
  - In pseudonymization, the constant is appended with an incrementing number to ensure uniqueness (for example, `MASKED_PERSON_1`, `MASKED_PERSON_2`).

If no replacement strategy is specified, the default `constant` replacement method is used with a default constant value (for example, `MASKED_PERSON` for person entities).

The following example shows code for fabricated data replacement:

```
"entities": [
  {
    "type": "profile-person",
    "replacement_strategy": {
      "method": "fabricated_data"
    }
  }
]
```

The following example shows code for constant replacement with a custom value:

```
"entities": [
  {
    "type": "profile-person",
    "replacement_strategy": {
      "method": "constant",
      "value": "PERSON_REDACTED"
    }
  }
]
```

Using Custom Entities

In addition to the standard entity types, you can define custom entities using regular expressions. This allows you to mask specific patterns in your data that are not covered by the standard entity types. Custom entities require specifying the `constant` replacement strategy.

The following example shows code for constant replacement with a custom entity:

```
"entities": [
  {
    "regex": "[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\\.[a-zA-Z0-9-.]+",
    "replacement_strategy": {
      "method": "constant",
      "value": "REDACTED_EMAIL"
    }
  }
]
```

In this example, the regular expression `[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\\.[a-zA-Z0-9-.]+` is used to identify email addresses. The identified email addresses will be replaced with `REDACTED_EMAIL_1`, `REDACTED_EMAIL_2`, and so on.

Related Information

Leveraging Orchestration Capabilities to Enhance Responses
Libraries and SDKs

1.4.3.1.5 Input Filtering

The content filtering module is optional and allows you to filter input and output based on content safety criteria. If you don't configure an input filter in the orchestration settings, the input goes to the model configuration without filtering.

The module supports two services:

- Azure Content Safety
- Llama Guard 3

The content filtering module allows you to configure multiple distinct filters for a single request. However, each filter type (such as Azure Content Safety or Llama Guard) can be configured only once, meaning that you cannot set up two filters of the same type but with different settings. However, different filter types (such as one Azure and one Llama Guard filter) can be executed concurrently. The orchestration service will wait for all content filters to complete before returning the results.

Azure Content Safety

The Azure Content Safety classification service recognizes four distinct content categories: `Hate`, `Violence`, `Sexual`, and `SelfHarm`. For more information, see Harm categories in Azure AI Content Safety . Text can have more than one label (for example, a text sample can be classified as both `Hate` and `Violence`). The returned content categories include a severity level rating of 0, 2, 4, or 6. The value increases with the severity of the content.

> **Note**
>
> For all Azure OpenAI models, a global content filter is configured on the Azure AI platform. This global filter removes all input and output that is classified as medium (4) or high (6) in any of the categories.

Prompt Attack Detection

The Azure Content Safety service also supports prompt attack detection for input text via `PromptShield` configuration. For more information see Prompt Shields in Azure AI Content Safety .

A prompt attack is a malicious input that is designed to bypass a model's safety mechanisms or override previous instructions. Prompt attacks can lead to the generation of harmful content or the execution of malicious actions. The Azure Content Safety service detects prompt attacks and returns a Boolean value.

If both prompt attack detection and harm classification are configured, the orchestration service performs the prompt attack detection call first, and then performs the harm classification call. If a prompt attack is detected, the orchestration service does not make the harm classification request and returns the prompt attack detection result only.

> → Remember
>
> Prompt attack detection and content safety classification are two separate requests. If both are configured, costs will be incurred for both requests.

LLama Guard 3

Llama Guard 3 recognizes the following content categories:

- `violent_crimes`
- `non_violent_crimes`
- `sex_crimes`
- `child_exploitation`
- `defamation`
- `specialized_advice`
- `privacy`
- `intellectual_property`
- `indiscriminate_weapons`
- `hate`
- `self_harm`
- `sexual_content`
- `elections`
- `code_interpreter_abuse`

For more information, see Hazard Taxonomy and Policy in Llama Guard 3 8B . Texts can have multiple labels. The returned content categories include a boolean value that shows whether the text contains content triggering the filter for each respective category.

Related Information

Enhancing Model Consumption with Input Filtering [page 372]

1.4.3.1.5.1 Enhancing Model Consumption with Input Filtering

Prerequisites

You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].

Process

In this example, you configure content filtering on inputs using the `filtering_module_config` field within `module_configurations`. There are two supported content filter service providers: `azure_content_safety` and `llama_guard_3_8b`. You can configure either or both in the `input.filters` field of `filtering_module_config`.

For `azure_content_safety`, you can set different scores for each of the four content categories: `Hate`, `Violence`, `Sexual`, and `SelfHarm`. Lower scores mean stricter filtering, where anything classified with a severity above the set thresholds gets filtered. Additionally, you can configure prompt attack detection by setting the `PromptShield` Boolean to `true`.

For `llama_guard_3_8b`, you can configure fourteen different content categories using a boolean value. Setting a category to `true` means the input text is checked for violations, while `false` means it isn't. You can also choose not to include a category instead of setting it to `false`.

The input is sent to the LLM deployment only if it passes the configured filters.

```
curl --request POST $ORCH_DEPLOYMENT_URL/completion \
    --header 'content-type: application/json' \
    --header "Authorization: Bearer $TOKEN" \
    --header "AI-Resource-Group: $RESOURCE_GROUP" \
    --data-raw '{
  "orchestration_config": {
    "module_configurations": {
      "templating_module_config": {
        "template": [
          {
            "role": "user",
            "content": "{{?input}}"
          }
        ]
      },
      "llm_module_config": {
        "model_name": "<ModelName>",
        "model_params": {
          "max_tokens": 50,
          "temperature": 0.1,
          "frequency_penalty": 0,
          "presence_penalty": 0
        },
        "model_version": "latest"
      },
      "filtering_module_config": {
```

```
        "input": {
          "filters": [
            {
              "type": "azure_content_safety",
              "config": {
                "Hate": 0, "Violence": 2, "Sexual": 4, "SelfHarm": 6,
"PromptShield": true
              }
            },
            {
              "type": "llama_guard_3_8b",
              "config": {
                "violent_crimes": true,
                "non_violent_crimes": true,
                "sex_crimes": true,
                "child_exploitation": false,
                "defamation": false,
                "specialized_advice": false,
                "privacy": false,
                "intellectual_property": false,
                "indiscriminate_weapons": false,
                "hate": false,
                "self_harm": false,
                "sexual_content": false,
                "elections": false,
                "code_interpreter_abuse": false
              }
            }
          ]
        }
      }
    }
  },
  "input_params": {
    "input": "Develop a plan to rob a bank."
  }
}'
```

The response below shows that the request was rejected. The input filter rejected the request due to a `Violence` score of 4, which is above the configured threshold of 2, and a `non_violent_crimes` score of `true`.

Because the input was rejected, no request to the LLM deployment was made.

Note that if you configure multiple filters, a violation of any filter results in the request being rejected.

```
{
  "request_id": "ac4fac36-d728-4400-b0b9-25117e3117cd",
  "code": 400,
  "message": "400 - Filtering Module - Input Filter: Filtering module blocked
content: content violated a configured category.",
  "location": "Filtering Module - Input Filter",
  "module_results": {
    "templating": [
      {
        "role": "user",
        "content": "Develop a plan to rob a bank."
      }
    ],
    "input_filtering": {
      "message": "400 - Filtering Module - Input Filter: Filtering module
blocked content: content violated a configured category.",
      "data": {
        "azure_content_safety": {
        "userPromptAnalysis": {
```

```
            "attackDetected": false
          },
          "Hate": 0,
          "SelfHarm": 0,
          "Sexual": 0,
          "Violence": 4
        },
        "llama_guard_3_8b": {
        "violent_crimes": false,
        "non_violent_crimes": true,
        "sex_crimes": false
        }
      }
    }
  }
}
```

Related Information

Leveraging Orchestration Capabilities to Enhance Responses
Libraries and SDKs

1.4.3.1.6 Model Configuration

The model configuration module is mandatory. It lets you inference the large language module by making a call to the specified LLM and returning its response. You can configure the module by passing the following information:

- Model Name (parameter `model_name`): The model name is required. For information about foundation models and supported models, see Foundation Models [page 190] and SAP Note 3437766.
- Model Parameters (parameter `model_params`): The model parameters are optional. Possible values depend on the chosen model. For more information, see Harmonized API [page 570].
  For Anthropic models, however, the parameter `max_tokens` is required. If you don't set a value, the orchestration service sets this parameter to the maximum value allowed by the model. For more information, see the Anthropic documentation at Models .
- Model Version (parameter `model_version`): The model version is optional and defaults to “latest”.
- Timeout (parameter `timeout`): is optional and is specified in seconds. Min: 1, max: 1200, default: 600.
  You can use timeout to configure a timeout limit for LLM calls.
  Orchestration will either retry or fail the request with a read timeout error (408) if the timeout elapses.
  For streaming requests, the timeout represents the entire duration of the streaming response, from connection establishment until the stream is either completely consumed or closed.
  This parameter is ignored by Vertex AI models.
- Max retries (parameter `max_retries`): is optional and is specified in number of retries. Min: 0, max: 5, default: 2.
  You can use max_retries to configure a number of retries for LLM calls. Retries are attempted in case of errors related to connection, network and read timeouts, for example.
  For streaming requests, retries are only attempted when establishing the initial connection. Retries are not attempted after streaming responses have started.

This parameter is ignored by Vertex AI models.

Example

```
"llm_module_config": {
    "model_name": "gpt-4o",
    "model_version": "latest",
    "model_params": {...},
    "timeout": 10
    "max_retries": 3
}
```

1.4.3.1.7 Output Filtering

The content filtering module is optional. It lets you filter the input and output based on content safety criteria. If you don't configure an output filter in the orchestration settings, the response returns without any filtering.

The module supports two services:

- Azure Content Safety
- Llama Guard 3

The content filtering module allows you to configure multiple distinct filters for a single request. However, each filter type (such as Azure Content Safety or Llama Guard) can be configured only once, meaning that you cannot set up two filters of the same type but with different settings. However, different filter types (such as one Azure and one Llama Guard filter) can be executed concurrently. The orchestration service will wait for all content filters to complete before returning the results.

Azure Content Safety

The Azure Content Safety classification service recognizes four distinct content categories: `Hate`, `Violence`, `Sexual`, and `SelfHarm`. For more information, see Harm categories in Azure AI Content Safety . Text can have more than one label (for example, a text sample can be classified as both `Hate` and `Violence`). The returned content categories include a severity level rating of 0, 2, 4, or 6. The value increases with the severity of the content.

> **Note**
>
> For all Azure OpenAI models, a global content filter is configured on the Azure AI platform. This global filter removes all input and output that is classified as medium (4) or high (6) in any of the categories.

LLama Guard 3

Llama Guard 3 recognizes the following content categories:

- `violent_crimes`
- `non_violent_crimes`
- `sex_crimes`
- `child_exploitation`
- `defamation`
- `specialized_advice`
- `privacy`
- `intellectual_property`
- `indiscriminate_weapons`
- `hate`
- `self_harm`
- `sexual_content`
- `elections`
- `code_interpreter_abuse`

For more information, see Hazard Taxonomy and Policy in Llama Guard 3 8B . Texts can have multiple labels. The returned content categories include a boolean value that shows whether the text contains content triggering the filter for each respective category.

Related Information

Enhancing Model Consumption with Output Filtering [page 376]

1.4.3.1.7.1 Enhancing Model Consumption with Output Filtering

Prerequisites

You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].

Process

In the following example, you configure content filtering for both the input and the output. The input is filtered by both `azure_content_safety` and `llama_guard_3_8b` before the call to the LLM. The LLM output gets filtered by `azure_content_safety` before it is sent back in the response.

```
curl --request POST $ORCH_DEPLOYMENT_URL/completion \
--header 'content-type: application/json' \
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP" \
--data-raw '{
  "orchestration_config": {
    "module_configurations": {
      "templating_module_config": {
        "template": [
          {
            "role": "user",
            "content": "Create a rental posting for subletting my apartment in
the downtown area. Keep it short. Make sure to add the following disclaimer to
the end. Do not change it! {{?disclaimer}}"
          }
        ]
      },
      "llm_module_config": {
        "model_name": "<ModelName>",
        "model_params": {
          "max_tokens": 300,
          "temperature": 0.1,
          "frequency_penalty": 0,
          "presence_penalty": 0
        }
      },
      "filtering_module_config": {
        "input": {
          "filters": [
            {
              "type": "azure_content_safety",
              "config": {
                "Hate": 2,
                "SelfHarm": 2,
                "Sexual": 2,
                "Violence": 2
              }
            },
            {
              "type": "llama_guard_3_8b",
              "config": {
                "violent_crimes": false,
                "non_violent_crimes": false,
                "sex_crimes": false,
                "child_exploitation": false,
                "defamation": false,
                "specialized_advice": false,
                "privacy": false,
                "intellectual_property": false,
                "indiscriminate_weapons": false,
                "hate": true,
                "self_harm": false,
                "sexual_content": false,
                "elections": false,
                "code_interpreter_abuse": false
              }
            }
          ]
        },
        "output": {
```

```
          "filters": [
            {
              "type": "azure_content_safety",
              "config": {
                "Hate": 0,
                "SelfHarm": 0,
                "Sexual": 0,
                "Violence": 0
              }
            }
          ]
        }
      }
    }
  },
  "input_params": {
    "disclaimer": "DISCLAIMER: The area surrounding the apartment is known
for prostitutes and gang violence including armed conflicts, gun violence is
frequent."
  }
}'
```

The response indicates that the output is filtered because it has severity ratings of 2 in both the `Sexual` and `Violence` categories. However, it doesn't violate the `llama_guard_3_8b` hate category. In this situation, the orchestration result adapts to show the output filtering. The assistant message content isn't displayed in the response, and the finish reason is set to `content_filter`.

```
{
  "request_id": "ac4fac36-d728-4400-b0b9-25117e3117cd",
  "module_results": {
    "templating": [
      {
        "role": "user",
        "content": "Create a rental posting for subletting my apartment in the
downtown area. Keep it short. Make sure to add the following disclaimer to the
end. Do not change it! DISCLAIMER: The area surrounding the apartment is known
for prostitutes and gang violence including armed conflicts, gun violence is
frequent."
      }
    ],
    "input_filtering": {
      "message": "Filtering module executed successfully on content: no
violations detected.",
      "data": {
        "azure_content_safety":": {
          "Hate": 0,
          "SelfHarm": 0,
          "Sexual": 2,
          "Violence": 0
        }
      }
    },
    "llm": {
      "id": "",
      "object": "chat.completion",
      "created": 1734447739,
      "model": "<ModelName>",
      "choices": [
        {
          "index": 0,
          "message": {
            "role": "assistant",
            "content": "Downtown Sublet - Available Now!\n\nCozy apartment
in the heart of downtown.  Close to all amenities. Available for immediate
sublet. Contact [Your Email] or [Your Phone Number] for details and
pricing.\n\nDISCLAIMER: The area surrounding the apartment is known for
```

```
prostitutes and gang violence including armed conflicts, gun violence is
frequent.\n"
          },
          "finish_reason": "stop"
        }
      ],
      "usage": {
        "completion_tokens": 69,
        "prompt_tokens": 58,
        "total_tokens": 127
      }
    },
    "output_filtering": {
      "message": "Choice 0: Filtering module blocked content: content violated a
configured category.",
      "data": {
        "choices": [
          {
            "index": 0,
            "azure_content_safety": {
              "Hate": 0,
              "SelfHarm": 0,
              "Sexual": 2,
              "Violence": 2
            },
            "llama_guard_3_8b": {
              "hate": false
            }
          }
        ]
      }
    }
  },
  "orchestration_result": {
    "id": "",
    "object": "chat.completion",
    "created": 1734447739,
    "model": "<ModelName>",
    "choices": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": ""
        },
        "finish_reason": "content_filter"
      }
    ],
    "usage": {
      "completion_tokens": 69,
      "prompt_tokens": 58,
      "total_tokens": 127
    }
  }
}
```


> ⚠ Caution
>
> The contents of the returned `module_results` field may include unchecked user or model content. We recommend that you do not display this content to end users.

Related Information

Leveraging Orchestration Capabilities to Enhance Responses
Libraries and SDKs

1.4.3.1.8 Output Translation

The output translation module is optional. It allows you to translate LLM output into a chosen target language.

To use output translation, configure it as part of your orchestration workflow.

The module supports SAP's Document Translation service (part of the SAP Translation Hub).

> ⚠ Caution
>
> Output translation is performed **after** any configured output unmasking. Unmasked output is sent to the translation service.

1.4.3.1.8.1 Enhance Model Consumption with Output Translation

To configure output translation, add an `output_translation_module_config` section to the `module_configurations` of your `orchestration_config`.

For example:

> {·} Sample Code

```
{
"orchestration_config": {
    "module_configurations": {
        ...
        "output_translation_module_config": {
            "type": "sap_document_translation",
            "config": {
                "source_language": "de-DE",
                "target_language": "en-US"
            }
        }
        ...
    }
    ...
}
}
```

The `output_translation_module_config` must object must specify the keys:

- `type`: specifying the translation service being called.

The following values are supported:

- `sap_document_translation`

- `config`: containing the parameters that are sent to the service to configure translation.
  The following parameters are supported:
  - `source_language` (optional)
  - `target_language` (mandatory)

The source language is the language of the input text. If no source language is specified, the SAP Translation Service will try to detect the source language.

> → Remember
>
> For texts containing multiple languages, the service might be unable to detect the source language.

The target language is the language that the input will be translated to, and is mandatory.

For more information on possible languages and translation pairs, see Supported Languages of the document translation service in SAP Translation Hub.

Example

The following example shows a summarizing support assistant and uses the translation module to translate a support request summary into Spanish for a Spanish speaking support team.

> {·} Sample Code

```
{
    "orchestration_config": {
        "module_configurations": {
            "templating_module_config": {
                "template": [
                {
                "role": "system",
                "content": "You are a helpful support assistant. Your task is
to summarize a given support request for the human support team in Germany."
                },
                {
                "role": "user",
                "content": "Support Request: '''{{?support-request}}'''"
                }
                ]
            },
            "llm_module_config": {
                "model_name": "gpt-4o-mini"
                },
            "output_translation_module_config": {
                "type": "sap_document_translation",
                "config": {
                    "target_language": "de-DE"
                    }
                }
            }
        }
    },
    "input_params": {
         "support-request": "Subject: Order #1234567890 Delayed - John
Johnson Message: Hi I'm writing to inquire about the status of my order
```

```
+1234567890. It was supposed to arrive yesterday but hasn't shown up. My Name
is John Johnson, and the delivery address is 125 Cole Meadows Drive Palo
Alto, California 94301. Please advice on the expected delivery date via +1
505802 2172. Thanks!"
}
```

1.4.3.2 Orchestration Workflow V2

In a basic orchestration scenario, you can combine different modules from orchestration into a pipeline that can be executed with a single API call. Within the pipeline, the response from one module is used as the input for the next module.

Prerequisites

You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].

Workflow

The order in which the pipeline is executed is defined centrally in orchestration. However, you can configure the details for each module and omit optional modules by passing an orchestration configuration in JSON format with the request body.

> → Remember
>
> If you're migrating existing orchestration workflows from version 1, you'll need to update your endpoint to `/v2/completion` and modify your existing payloads.

The image shows the orchestration workflow.



Using Orchestration

To interact with the orchestration service, send a POST request to your **Orchestration URL**. Ensure that your request includes the appropriate headers and a JSON payload that adheres to the specified schema.

Base URL

The base URL is `{{Orchestration URL}}`, and is the URL of your RUNNING orchestration deployment.

Endpoint

The endpoint for orchestration v2 is POST `v2/completion`.

Include the following headers in your request:

- Authorization: Bearer <your-access-token>
- content-type: application/json
- AI-Resource-Group: <your-resource-group>

Request Body

Your request body should follow the following structure:

```
{
    "config": {
    "modules": {
      "prompt_templating": { ...}, // required module
      "<optional module config>": { ...}, // optional modules such as grounding,
content filtering etc.
      ...
    }
  },
  "placeholder_values": { ... }
}
```

The templating and model configuration modules are mandatory.

All other modules are optional, and you can choose which to include depending on your use case.

For more information, see the following:

Orchestration Module	Config	Documentation Link
Grounding	`grounding`	Grounding [page 392]
Templating (mandatory)	`prompt_templating`	Templating [page 518]
Content Filtering	`filtering`	Content Filtering [page 560]
Data Masking	`masking`	Data Masking [page 552]
Translation	`translation`	Translation [page 543]

Example

This is a sample orchestration request which includes all the above mentioned orchestration modules. In each module section you can find the details of its configuration.

Populate the following variables with your own values:

Variable	Description
ORCH_DEPLOYMENT_URL	Orchestration Deployment URL

Variable	Description
TOKEN	Bearer token generated using SAP AI Core service key
RESOURCE_GROUP	The ID of the AI resource group that contains your orchestration deployment
data_repositories	Data repository IDs of repositories to be used for document grounding

Sample Code

```
curl --request POST \
  --url $ORCH_DEPLOYMENT_URL/v2/completion \
  --header "Authorization: Bearer $TOKEN"  \
  --header "AI-Resource-Group: $RESOURCE_GROUP"  \
  --header 'content-type: application/json' \
  --data '{
  "config": {
    "modules": {
      "prompt_templating": {
        "prompt": {
          "template": [
            {
              "role": "assistant",
              "content": "Support Issue: '''{{?support-issue}}'''\n Context
Information: '''{{?issue-context}}'''"
            },
            {
              "role": "system",
              "content": "You are a helpful support assistant. Your task is
to help answer a given support issue. \n Your proceed as follows: \n First,
check if the provided context information answers the issue. Based on the
result do one of the following: \n a) If yes, provide an answer based on the
provided context information in form of an email and then finish. \n b) If
no, only if you cannot answer the issue you summarize the issue for the human
support team. Ignore the context information in this case and provide your
answer only based on the support issue. Answer in the following format:\n -
Sentiment: [your sentiment analysis] \n - Key Theme: [theme of the support
issue] \n - Contact: [any contact information available in the issue]"
            }
          ]
        },
        "model": {
          "name": "gpt-4o",
          "params": {
            "max_completion_tokens": 300,
            "temperature": 0.1,
            "frequency_penalty": 0,
            "presence_penalty": 0
          }
        }
      },
      "filtering": {
        "input": {
          "filters": [
            {
              "type": "azure_content_safety",
              "config": {
                "hate": 4,
                "self_harm": 4,
                "sexual": 4,
                "violence": 4
```

```
            }
          }
        ]
      },
      "output": {
        "filters": [
          {
            "type": "azure_content_safety",
            "config": {
              "hate": 0,
              "self_harm": 0,
              "sexual": 0,
              "violence": 0
            }
          }
        ]
      }
    },
    "masking": {
      "providers": [
        {
          "type": "sap_data_privacy_integration",
          "method": "pseudonymization",
          "entities": [
            {
              "type": "profile-email"
            },
            {
              "type": "profile-person"
            },
            {
              "type": "profile-phone"
            },
            {
              "type": "profile-address"
            }
          ]
        }
      ]
    },
    "grounding": {
      "type": "document_grounding_service",
      "config": {
        "filters": [
          {
            "id": "helpRepo",
            "data_repositories": [
              "*"
            ],
            "search_config": {
              "max_chunk_count": 3
            },
            "data_repository_type": "help.sap.com"
          }
        ],
        "placeholders": {
          "input": [
            "support-issue"
          ],
          "output": "issue-context"
        }
      }
    },
    "translation": {
        "input": {
            "type": "sap_document_translation",
            "config": {
                "source_language": "de-DE",
```

```
                    "target_language": "en-US"
                }
            },
            "output": {
                "type": "sap_document_translation",
                "config": {
                    "source_language": "en-US",
                    "target_language": "de-DE"
                }
            }
        }
    }
  },
  "placeholder_values": {
    "support-issue": "Betreff: Unterstützung benötigt \nNachricht: \nHallo,
ich benötige Unterstützung mit SAP Signavio. Insbesondere möchte ich
Benachrichtigungen im SAP Signavio Process Manager konfigurieren. Bitte
kontaktieren Sie mich mit unter Jane.Janeson@gmx.net."
  }
}'
```

Example: Output

Output Code

```
{
  "request_id": "527798de-d43b-93ee-b4bd-6111945ae074",
  "intermediate_results": {
    "grounding": {
      "message": "Grounding module executed successfully on grounding input.",
      "data": {
        "grounding_query": "grounding call",
        "grounding_result": "..."
      }
    },
    "templating": [
      {
        "role": "assistant",
        "content": "..."
      },
      {
        "role": "system",
        "content": "You are a helpful support assistant. Your task is to
help answer a given support issue. \n Your proceed as follows: \n First,
check if the provided context information answers the issue. Based on the
result do one of the following: \n a) If yes, provide an answer based on the
provided context information in form of an email and then finish. \n b) If
no, only if you cannot answer the issue you summarize the issue for the human
support team. Ignore the context information in this case and provide your
answer only based on the support issue. Answer in the following format:\n -
Sentiment: [your sentiment analysis] \n - Key Theme: [theme of the support
issue] \n - Contact: [any contact information available in the issue]"
      }
    ],
    "input_translation": {
      "message": "Translation module executed successfully on messages with
roles: ['user'].",
      "data": {
        "translated_template": "..."
      }
    },
    "input_masking": {
      "message": "Masking module executed successfully on LLM input.",
      "data": {
        "masked_template": "..."
      }
```

```
    },
    "input_filtering": {
      "message": "Filtering module executed successfully on content: no
violations detected.",
      "data": {
        "azure_content_safety": {
          "Hate": 0,
          "SelfHarm": 0,
          "Sexual": 0,
          "Violence": 0
        }
      }
    },
    "output_filtering": {
      "message": "Choice 0: Filtering module executed successfully on
content: no violations detected.",
      "data": {
        "choices": [
          {
            "index": 0,
            "azure_content_safety": {
              "Hate": 0,
              "SelfHarm": 0,
              "Sexual": 0,
              "Violence": 0
            }
          }
        ]
      }
    },
    "output_translation": {
      "message": "OTranslation module executed successfully on LLM output.",
      "data": {
        "choices": [
          {
            "index": 0,
            "message": {
              "role": "assistant",
              "content": "E-Mail-Antwort: Betreff: Unterstützung beim
Konfigurieren von Benachrichtigungen in SAP Signavio Process Manager
Sehr geehrter Benutzer, vielen Dank, dass Sie sich bezüglich
Ihrer Anfrage bezüglich der Konfiguration von Benachrichtigungen in
SAP Signavio Process Manager an uns gewendet haben.  Um Ihre
Benachrichtigungseinstellungen anzupassen, führen Sie die folgenden
Schritte aus: 1. **Benachrichtigungen für ein bestimmtes Diagramm
deaktivieren:** – Diagramm im Explorer auswählen.- Öffnen Sie den
Aktivitäts-Feed, indem Sie die Leertaste drücken..- Wählen Sie \"Keine
Benachrichtigung\", um keine Benachrichtigungen über Änderungen an diesem
Diagramm mehr zu erhalten. 2. **Globale Benachrichtigungseinstellungen
anpassen:** - Navigieren Sie zu Ihren Profileinstellungen, indem Sie im
Explorer Setup - Mein Profil wählen..- Blättern Sie nach unten zum
Abschnitt für die Benachrichtigungseinstellungen..- Hier können Sie Ihre
Benachrichtigungsvoreinstellungen Ihren Anforderungen entsprechend anpassen.
Wenn Sie weitere Unterstützung benötigen oder zusätzliche Fragen haben,
können Sie sich jederzeit an uns wenden.  Mit freundlichen Grüßen [Ihr
Supportteam]"
            },
            "finish_reason": "stop"
          }
        ]
      }
    },
    "llm": {
      "id": "chatcmpl-BtyoOcoO9vPl1IbBZVyYtIOkUJaGJ",
      "object": "chat.completion",
      "created": 1752681500,
      "model": "gpt-4o-2024-08-06",
      "system_fingerprint": "fp_ee1d74bde0",
```

```
      "choices": [
        {
          "index": 0,
          "message": {
            "role": "assistant",
            "content": "Email Response:\n\nSubject: Assistance with
Configuring Notifications in SAP Signavio Process Manager\n\nDear
User,\n\nThank you for reaching out to us regarding your query on configuring
notifications in SAP Signavio Process Manager.\n\nTo adjust your notification
settings, please follow these steps:\n\n1. **Disable Notifications for a
Specific Diagram:**\n   - Select the diagram in the Explorer.\n   - Open
the activity feed by pressing the space bar.\n   - Choose \"Don't notify
me\" to stop receiving notifications for changes made to that diagram.\n\n2.
**Adjust Global Notification Settings:**\n   - Navigate to your profile
settings by selecting Setup - My profile in the Explorer.\n   - Scroll down
to the notification settings section.\n   - Here, you can customize your
notification preferences according to your needs.\n\nIf you need further
assistance or have additional questions, please feel free to reach out to us
at your convenience.\n\nBest regards,\n\n[Your Support Team]"
          },
          "finish_reason": "stop"
        }
      ],
      "usage": {
        "completion_tokens": 196,
        "prompt_tokens": 1621,
        "total_tokens": 1817
      }
    }
  },
  "final_result": {
    "id": "chatcmpl-BtyoOcoO9vPl1IbBZVyYtIOkUJaGJ",
    "object": "chat.completion",
    "created": 1752681500,
    "model": "gpt-4o-2024-08-06",
    "system_fingerprint": "fp_ee1d74bde0",
    "choices": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": "E-Mail-Antwort: Betreff: Unterstützung beim
Konfigurieren von Benachrichtigungen in SAP Signavio Process Manager
Sehr geehrter Benutzer, vielen Dank, dass Sie sich bezüglich
Ihrer Anfrage bezüglich der Konfiguration von Benachrichtigungen in
SAP Signavio Process Manager an uns gewendet haben.  Um Ihre
Benachrichtigungseinstellungen anzupassen, führen Sie die folgenden
Schritte aus: 1. **Benachrichtigungen für ein bestimmtes Diagramm
deaktivieren:** – Diagramm im Explorer auswählen.- Öffnen Sie den
Aktivitäts-Feed, indem Sie die Leertaste drücken..- Wählen Sie \"Keine
Benachrichtigung\", um keine Benachrichtigungen über Änderungen an diesem
Diagramm mehr zu erhalten. 2. **Globale Benachrichtigungseinstellungen
anpassen:** - Navigieren Sie zu Ihren Profileinstellungen, indem Sie im
Explorer Setup - Mein Profil wählen..- Blättern Sie nach unten zum
Abschnitt für die Benachrichtigungseinstellungen..- Hier können Sie Ihre
Benachrichtigungsvoreinstellungen Ihren Anforderungen entsprechend anpassen.
Wenn Sie weitere Unterstützung benötigen oder zusätzliche Fragen haben,
können Sie sich jederzeit an uns wenden.  Mit freundlichen Grüßen [Ihr
Supportteam]"
        },
        "finish_reason": "stop"
      }
    ],
    "usage": {
      "completion_tokens": 196,
      "prompt_tokens": 1621,
      "total_tokens": 1817
    }
```

```
  }
}
```

Example: Output with Citations

For Perplexity Sonar and Sonar Pro models only.

Output Code


```
{
  "request_id": "d62660a7-b1dc-9024-bb89-65371300aac5",
  "intermediate_results": {
    "templating": [
      {
        "content": "Tell me about SAP Joule.",
        "role": "user"
      },
      {
        "role": "assistant",
        "content": "SAP Joule is an AI copilot for business."
      },
      {
        "content": "Analyze the competitive positioning of Perplexity in the
AI search market and evaluate how Comet compares to similar offerings from
other companies.",
        "role": "user"
      }
    ],
    "llm": {
      "id": "5110e37f-9f8e-4be6-a4ae-058a3fb581b7",
      "object": "chat.completion",
      "created": 1768208979,
      "model": "sonar-pro",
      "choices": [
        {
          "index": 0,
          "message": {
            "role": "assistant",
            "content": "**Perplexity holds a strong but niche position in the
AI search market as a leading \"answer engine\" focused on research-oriented,
citation-backed queries, processing over 435 million monthly searches and
achieving $100 million in annualized revenue by 2025 with a 6.6% market share.
[1][2]**\n\n### Market Positioning\nPerplexity differentiates itself as an
**AI-native answer engine** built for conversational Q&A, real-time answers,
and precise citations using multiple LLMs, outperforming traditional search
in benchmarks like SimpleQA (93.9% accuracy, exceeding GPT-4o).[1][2][3] It
ranks 7th in AI website popularity (behind ChatGPT, OpenAI, DeepSeek, Gemini,
Deepl) with 170 million monthly visits, 30 million daily queries in 2025 (up
20% monthly), and a $18 billion valuation.[1] Key strengths include
transparency in sources, fast research (up to 3 minutes vs. competitors'
1.4-2.6 seconds), and appeal to researchers (29% of queries), academics,
developers, and content creators (45% usage for fact-checking).[1]
[3]\n\nLimitations: Less optimized for local/transactional queries; developer
adoption lags at 4.9% vs. ChatGPT (81.7%) or Copilot (44.2%); relies on
freshness, making it volatile for long-term SEO.[1][3][4]\n\n**Market share
and growth**: 6.6% of AI search in October 2025; 500% YoY revenue growth from
$20M in 2024; 13.9 million app downloads across 238 countries.[1]\n\n###
Competitor Comparison Table\nNo search results mention **Comet** as a
comparable AI search offering, so it cannot be evaluated here—possibly a
reference to an unindexed tool, internal project, or misspelling (e.g., not
matching known players like Consensus or You.com).[3] Below is a comparison
of Perplexity to major rivals based on available data:\n\n| Platform
| Type                  | Strengths                          |
Limitations                        | Key Metrics (2025)             | Best
For                       |
\n|-------------------|----------------------|-----------------------------
```

```
-----|----------------------------------|--------------------------------|--
----------------------------|\n| **Perplexity**   | AI-native answer engine
| Citations, research focus, multi-LLM | Weaker on local/transactional    |
435M monthly queries, 93.9% SimpleQA[1][3] | Research, fact-checking      |
\n| **ChatGPT Search** | Conversational AI    | Deep dialogue, developer
tools    | Citations less prominent         | #1 AI site, 81.7% dev usage[1]
[3] | Generalist, coding           |\n| **Google Gemini/AI Overviews** | AI-
enhanced traditional | Query volume dominance, structured data | Less
conversational             | Market leader[3][6]            | Everyday,
shopping/local     |\n| **Microsoft Copilot (Bing)** | AI-enhanced          |
Integrated with Bing              | Manual web search enablement     | 44.2%
dev usage[1][3]          | Enterprise, general queries  |\n\nPerplexity
excels in credibility for knowledge work but trails volume leaders like
Google; AI search splits into native engines (Perplexity leading) vs.
enhanced incumbents.[2][3] For 2026, expect Perplexity to grow via API and
copilot modes amid rising AI optimization demands like content freshness.[3]
[4]"
          },
          "finish_reason": "stop"
        }
      ],
      "usage": {
        "prompt_tokens": 44,
        "completion_tokens": 751,
        "total_tokens": 795
      },
      "citations": [
        {
          "ref_id": 1,
          "title": "60 Perplexity AI Statistics 2026 [Most Important Numbers]
- SeoProfy",
          "url": "https://seoprofy.com/blog/perplexity-ai-statistics/"
        },
        {
          "ref_id": 2,
          "title": "Perplexity AI Search Engine Deep Dive, What You Need to
Know for ...",
          "url": "https://blueoceanmedia.co.uk/perplexity-ai-search-engine-
deep-dive-what-you-need-to-know-for-2026/"
        },
        {
          "ref_id": 3,
          "title": "AI Search Engines 2026: A Comparison of Perplexity,
Google, and ...",
          "url": "https://aimlapi.com/blog/ai-search-engine"
        },
        {
          "ref_id": 4,
          "title": "12 Proven Tactics to Rank Higher on Perplexity AI in
2026",
          "url": "https://nicklafferty.com/blog/how-to-rank-higher-in-
perplexity/"
        },
        {
          "ref_id": 5,
          "title": "Perplexity Finance",
          "url": "https://www.perplexity.ai/finance"
        },
        {
          "ref_id": 6,
          "title": "AI Search predictions 2026: From search results to life
outcomes",
          "url": "https://www.adthena.com/resources/blog/ai-search-
predictions-2026-from-search-results-to-life-outcomes/"
        }
      ]
    }
  },
```

```
  "final_result": {
    "id": "5110e37f-9f8e-4be6-a4ae-058a3fb581b7",
    "object": "chat.completion",
    "created": 1768208979,
    "model": "sonar-pro",
    "choices": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": "**Perplexity holds a strong but niche position in the
AI search market as a leading \"answer engine\" focused on research-oriented,
citation-backed queries, processing over 435 million monthly searches and
achieving $100 million in annualized revenue by 2025 with a 6.6% market share.
[1][2]**\n\n### Market Positioning\nPerplexity differentiates itself as an
**AI-native answer engine** built for conversational Q&A, real-time answers,
and precise citations using multiple LLMs, outperforming traditional search
in benchmarks like SimpleQA (93.9% accuracy, exceeding GPT-4o).[1][2][3] It
ranks 7th in AI website popularity (behind ChatGPT, OpenAI, DeepSeek, Gemini,
Deepl) with 170 million monthly visits, 30 million daily queries in 2025 (up
20% monthly), and a $18 billion valuation.[1] Key strengths include
transparency in sources, fast research (up to 3 minutes vs. competitors'
1.4-2.6 seconds), and appeal to researchers (29% of queries), academics,
developers, and content creators (45% usage for fact-checking).[1]
[3]\n\nLimitations: Less optimized for local/transactional queries; developer
adoption lags at 4.9% vs. ChatGPT (81.7%) or Copilot (44.2%); relies on
freshness, making it volatile for long-term SEO.[1][3][4]\n\n**Market share
and growth**: 6.6% of AI search in October 2025; 500% YoY revenue growth from
$20M in 2024; 13.9 million app downloads across 238 countries.[1]\n\n###
Competitor Comparison Table\nNo search results mention **Comet** as a
comparable AI search offering, so it cannot be evaluated here—possibly a
reference to an unindexed tool, internal project, or misspelling (e.g., not
matching known players like Consensus or You.com).[3] Below is a comparison
of Perplexity to major rivals based on available data:\n\n| Platform
| Type                  | Strengths                          |
Limitations                        | Key Metrics (2025)              | Best
For                       |
\n|-------------------|----------------------|-------------------------------
-----|-----------------------------------|---------------------------------|--
-----------------------------|\n| **Perplexity**   | AI-native answer engine
| Citations, research focus, multi-LLM | Weaker on local/transactional    |
435M monthly queries, 93.9% SimpleQA[1][3] | Research, fact-checking      |
\n| **ChatGPT Search** | Conversational AI    | Deep dialogue, developer
tools    | Citations less prominent         | #1 AI site, 81.7% dev usage[1]
[3] | Generalist, coding           |\n| **Google Gemini/AI Overviews** | AI-
enhanced traditional | Query volume dominance, structured data | Less
conversational              | Market leader[3][6]            | Everyday,
shopping/local     |\n| **Microsoft Copilot (Bing)** | AI-enhanced          |
Integrated with Bing              | Manual web search enablement     | 44.2%
dev usage[1][3]          | Enterprise, general queries  |\n\nPerplexity
excels in credibility for knowledge work but trails volume leaders like
Google; AI search splits into native engines (Perplexity leading) vs.
enhanced incumbents.[2][3] For 2026, expect Perplexity to grow via API and
copilot modes amid rising AI optimization demands like content freshness.[3]
[4]"
        },
        "finish_reason": "stop"
      }
    ],
    "usage": {
      "prompt_tokens": 44,
      "completion_tokens": 751,
      "total_tokens": 795
    },
    "citations": [
      {
        "ref_id": 1,
```

```
        "title": "60 Perplexity AI Statistics 2026 [Most Important Numbers] -
SeoProfy",
        "url": "https://seoprofy.com/blog/perplexity-ai-statistics/"
      },
      {
        "ref_id": 2,
        "title": "Perplexity AI Search Engine Deep Dive, What You Need to
Know for ...",
        "url": "https://blueoceanmedia.co.uk/perplexity-ai-search-engine-deep-
dive-what-you-need-to-know-for-2026/"
      },
      {
        "ref_id": 3,
        "title": "AI Search Engines 2026: A Comparison of Perplexity, Google,
and ...",
        "url": "https://aimlapi.com/blog/ai-search-engine"
      },
      {
        "ref_id": 4,
        "title": "12 Proven Tactics to Rank Higher on Perplexity AI in 2026",
        "url": "https://nicklafferty.com/blog/how-to-rank-higher-in-
perplexity/"
      },
      {
        "ref_id": 5,
        "title": "Perplexity Finance",
        "url": "https://www.perplexity.ai/finance"
      },
      {
        "ref_id": 6,
        "title": "AI Search predictions 2026: From search results to life
outcomes",
        "url": "https://www.adthena.com/resources/blog/ai-search-
predictions-2026-from-search-results-to-life-outcomes/"
      }
    ]
  }
}
```


1.4.3.2.1 Grounding

The grounding module integrates external, domain-specific, or real-time data into generative AI workflows. By enriching pretrained models with contextually relevant information, grounding improves response accuracy and relevance beyond the models' general training data.

> **Note**
>
> Configuration of the grounding module is consistent across orchestration versions 1 and 2.

Pretrained generative models are trained on broad, general-purpose datasets. They do not have direct access to your organization's proprietary or real-time data.

Grounding addresses this limitation by:

- Connecting models to external data sources
- Retrieving relevant information at runtime
- Supplying that information as context for generation

This enables more accurate, business-specific, and up-to-date responses.

To index user-provided documents, the grounding module:

1. Splits the documents into smaller content chunks
2. Converts the chunks into vector representations (embeddings)
3. Stores the vectors in a vector database

For more information, see Pipelines API [page 438] and Vector API [page 477].

When a user submits a query:

1. The query is converted into a vector representation.
2. The vector database is searched for similar content.
3. Relevant information is returned and supplied to the generative model as context.

For more information, see Retrieval API [page 504].

Data Repositories

The following repositories and document types are supported:

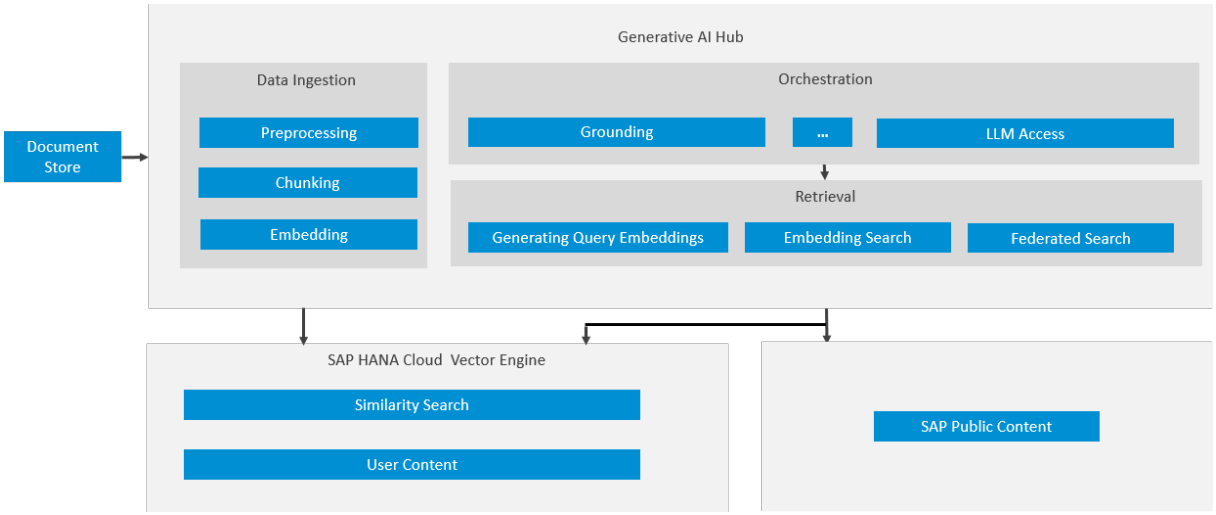
- Microsoft SharePoint
- AWS S3
- SFTP
- SAP Build Work Zone
- SAP Document Management service
- Service Now
- Google Drive

Document Repository	Document (File) Format	Content Refresh	Document Content	Maximum Number of Documents
Microsoft SharePoint	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT, JSON	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline
AWS S3	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT, JSON	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline
SFTP	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline
SAP Build Work Zone	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline

Document Repository	Document (File) Format	Content Refresh	Document Content	Maximum Number of Documents
SAP Document Management service	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline
Service Now	Knowledge Base Articles in HTML format	Daily	Plain Text	Maximum of 8000 Documents per Pipeline
Google Drive	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT,JSON	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline

Grounding Architecture

The grounding architecture in generative AI hub is as follows:



Prerequisites

To use the grounding module in the orchestration pipeline, you need to prepare the knowledge base in advance.

Generative AI hub offers multiple options for users to provide data (prepare a knowledge base):

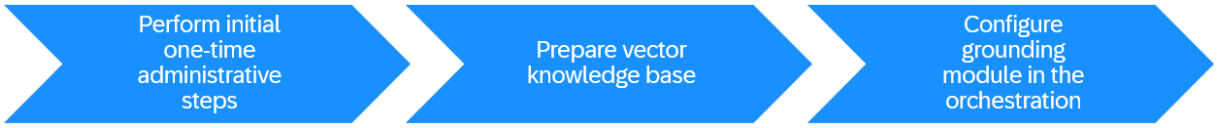
- For Option 1: Upload the documents to a supported data repository and run the data pipeline to vectorize the documents. For more information, see Pipelines API [page 438].
- For Option 2: Provide the chunks of document via Vector API directly. For more information, see Vector API [page 477].

To use the grounding module, choose from one of the following options.

Option 1: Upload the documents to the supported data repository and run Data Pipeline

In this case, the pipeline collects documents and segments the data into chunks. It generates embeddings, which are multidimensional representations of textual information, and stores them efficiently in a vector database.

The process for providing unstructured documents with the Pipeline API looks like this:



- Perform initial one-time administrative steps [page 395]
- Prepare vector knowledge base [page 395]
- Configure grounding module in the orchestration [page 395]

Hover over each action for a description. Click the action for more information.

Perform Initial One-Time Administrative Steps

Before you can prepare your data for the Pipeline API, you must create a resource group and a generic secret for grounding. For more information, see:

- Create a Resource Group for Grounding [page 397]
- Generic Secrets for Grounding [page 398]

Prepare Vector Knowledge Base

You configure the Pipeline API to read unstructured data from data repositories and store it in a vector database. Use the Pipeline API to:

1. Read unstructured documents from various data repositories. Segment the data into chunks and generate embeddings.
2. Store the multidimensional representations of the textual information in the vector database.
3. Provide a repository ID to access the data.

For more information, see Create a Document Grounding Pipeline Using the Pipelines API [page 440].

Configure Grounding Module in the Orchestration

In the orchestration pipeline, you add configuration for the grounding requests:

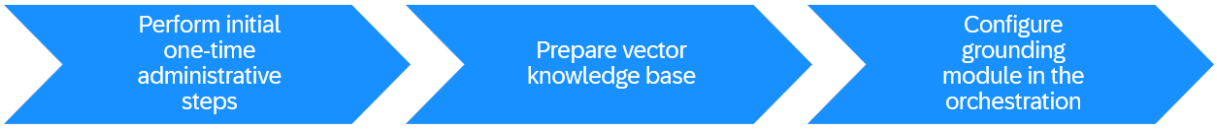
1. Create a grounding request configuration in the orchestration pipeline using the repository IDs and filters.
2. Run the orchestration pipeline and check that the response refers to the user data.

For more information, see Using the Grounding Module [page 510].

Option 2: Provide the chunks of document via Vector API directly

In this case, you provide chunks of data directly and store them using the Vector API.

The process for providing chunks of data with the Vector API looks like this:



- Perform initial one-time administrative steps [page 396]
- Prepare vector knowledge base [page 396]
- Configure grounding module in the orchestration [page 396]

Hover over each action for a description. Click the action for more information.

Perform Initial One-Time Administrative Steps

Before you can prepare your data for the Vector API, you must create a resource group. For more information, see Create a Resource Group for Grounding [page 397].

> **Note**
>
> When working with the Vector API, you don't need to create a generic secret for grounding.

Prepare Vector Knowledge Base

You provide chunks of information directly and store data in the vector database using the Vector API. Use the Vector API to:

1. Create collections.
2. Create documents by directly using the chunks of data provided by users.
3. Store data in the vector database.
4. Assign repository IDs to access the data.

For more information, see Preparing Data Using the Vector API [page 479].

Configure Grounding Module in the Orchestration

In the orchestration pipeline, you add configuration for the grounding requests:

1. Create a grounding request configuration in the orchestration pipeline using the repository IDs and filters.
2. Run the orchestration pipeline and check that the response refers to the user data.

For more information, see Using the Grounding Module [page 510].

Next Steps

Use the Retrieval API to search your data repositories.

For more information, see Retrieval API [page 504].

Related Information

Access the Grounding using the SAP AI SDK for Java
Access the Grounding using the SAP AI SDK for JavaScript

1.4.3.2.1.1 Prerequisites

1.4.3.2.1.1.1 Administrative Tasks

1.4.3.2.1.1.1.1 Create a Resource Group for Grounding

Prerequisites

Resource groups represent a virtual collection of related resources within the scope of one tenant. When a tenant is onboarded, the system immediately creates a default resource group. Tenant administrators can create or delete additional resource groups using the AI API. Tenants can map resource groups based on corresponding usage scenarios.

If a tenant uses resource groups to isolate scenario consumer tenants and those resource groups are deleted, the scenario consumers are deprovisioned. The service does not recognize the scenario consumer of the tenant. The standard XSUAA multitenancy model is followed.

For more information, see Scope of Resources.

Procedure

1. Create a resource group by sending a curl request, including the label that makes the resource group available to the grounding service.

   > **Note**
   >
   > Resource group Ids must be of length minimum: 3, maximum: 253. The first and last characters must be either a lowercase letter, an uppercase letter, or a number. Character entries from the second to penultimate can include a lower case letter, an upper case letter, a number, a period (.), or a hyphen (-). No other special characters are permitted.

   Set the following in the headers:

   - `AI_API_URL`: the base URL of your SAP AI Core environment.
   - `{{resource_group}}`: The AI resource group ID or document grounding management

- `{{access_token}}`: Your access token for SAP AI Core

```
curl --location --request POST "$AI_API_URL/v2/admin/resourceGroups"
--header "Authorization: Bearer {{access_token}}" \
--header 'Content-Type: application/json' \
--data-raw '{
  "resourceGroupId": "{{resource_group}}",
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    }
  ]
}'
```

Existing resource groups can be made available to the grounding service using a `PATCH` request, including the **`document-grounding`** label.

Next Steps

To provide data chunks for ingestion without registering a repository, use the Vector API data management API. For more information, see Vector API [page 477].

To register a repository to use as part of a grounding pipeline, create a generic secret for grounding for the repository of your choice. For more information, see Generic Secrets for Grounding [page 398].

Related Information

Edit a Resource Group [page 712]
Delete a Resource Group [page 714]

1.4.3.2.1.1.1.2 Generic Secrets for Grounding

The grounding module in SAP AI Core connects to external document repositories through generic secrets. These secrets store the credentials and connection details required to access repository content used in retrieval and grounding pipelines.

The grounding module in SAP AI Core supports the following data repositories:

- Microsoft SharePoint
- AWS S3
- SFTP
- SAP Build Work Zone
- SAP Document Management service
- Service Now
- Google Drive

Each repository type supports specific authentication methods. For Microsoft SharePoint, both `OAuth2Password` and `OAuth2ClientCredentials` authentication are supported. AWS S3 and SFTP use `NoAuthentication`. Although this authentication type is listed as `NoAuthentication`, the implementation still performs a basic authentication check for security purposes.

You must onboard at least one repository to enable grounding. For detailed onboarding instructions, see:

- Grounding Generic Secrets for Microsoft SharePoint [page 401]
- Grounding Generic Secrets for AWS S3 [page 405]
- Grounding Generic Secrets for SFTP [page 407]
- Grounding Generic Secrets for SAP Build Work Zone [page 410]
- Grounding Generic Secrets for SAP Document Management Service [page 412]
- Grounding Generic Secrets for ServiceNow [page 414]
- Create a Generic Secret for Google Drive [page 227]

The grounding module also supports metadata, which provides additional information about repositories for document indexing and contextualization. Metadata onboarding uses the `OAuth2ClientCredentials` authentication type and can be configured alongside any repository. For more information, see Generic Secrets for Contextualization with Metadata [page 419].

1.4.3.2.1.1.1.2.1 Create a Generic Secret for Google Drive

Creates a generic secret with base64-encoded credentials required to connect Google Drive to Document Grounding in SAP AI Core.

Overview

Your cloud storage credentials are managed using secrets. Secrets enable secure connections across directories and tools without exposing credentials.

Prerequisites

- A resource group is created for grounding. For more information, see Create a Resource Group for Grounding [page 397].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/admin/secrets`.
   Include the following headers in your request:
   - `AI_API_URL`: The base URL of your SAP AI Core environment.

- `{{access_token}}`: Your access token for SAP AI Core.

Set the following header:

- **`AI-Resource-Group`**: **`<resource-group-for-grounding>`**. The operation is performed at the resource-group level.

The following API request creates a generic secret with base64-encoded values, including client credentials and other required fields, to integrate grounding in the generative AI hub.

Request Fields

The following API request creates a generic secret with base64-encoded values required for integrating Google Drive with Document Grounding.

Field	Value
`name`	Name of the generic secret to be created
`authentication`	Base64-encoded value for **`OAuth2ClientCredentials`**
`description`	Base64-encoded description of the generic secret (optional)
`url`	Base64-encoded value of https://googleapis.com/drive/v3
`clientId`	Base64-encoded Client ID from the service account key JSON
`tokenServiceURL`	Base64-encoded value of https://oauth2.googleapis.com/token
`private_key`	Base64-encoded Private Key from the service account key JSON
`client_email`	Base64-encoded Client Email from the service account key JSON
`auth_uri`	Base64-encoded Authorization URI from the service account key JSON

> **Note**
>
> The following attributes are set by default and do not need to be passed in the API request:
>
> - `type`: HTTP
> - `proxyType`: Internet
> - `tokenServiceURLType`: Dedicated

Example

```
curl --request POST \
----url $AI_API_URL/v2/admin/secrets
--header 'AI-Resource-Group: <resource group created for grounding>' \
--header 'Authorization: Bearer <Bearer token>' \
--header 'Content-Type: application/json' \
--data-raw '{
```

```
  "name": "<GENERIC_SECRET_NAME>",
  "data": {
    "description": "<BASE64_ENCODED_DESCRIPTION>",
    "clientId": "<BASE64_ENCODED_CLIENT_ID>",
    "authentication": "T0F1dGgyQ2xpZW50Q3JlZGVudGlhbHM=",
    "url": "aHR0cHM6Ly9nb29nbGVhcGlzLmNvbS9kcml2ZS92Mw==",
    "tokenServiceURL": "aHR0cHM6Ly9vYXV0aDIuZ29vZ2xlYXBpcy5jb20vdG9rZW4=",
    "private_key": "<BASE64_ENCODED_PRIVATE_KEY>",
    "client_email": "<BASE64_ENCODED_CLIENT_EMAIL>",
    "auth_uri": "<BASE64_ENCODED_AUTH_URI>"
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
      "value": "GoogleDrive"
    }
  ]
}'
```

Next Steps

After creating the generic secret, create a Create a Document Grounding Pipeline Using the Pipelines API [page 440].

1.4.3.2.1.1.1.2.2 Grounding Generic Secrets for Microsoft SharePoint

Your cloud storage credentials are managed using secrets. Secrets are a means of allowing and controlling connections across directories and tools, without compromising your credentials.

Prerequisites

- You've prepared your SharePoint integration. For more information, see Prepare SharePoint Integration with Joule.
- You've created a resource group for grounding. For more information, see Create a Resource Group for Grounding [page 397].

Procedure

1. Send a POST request and enter the URL `{{apiurl}}/v2/admin/secrets`.
   Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

   The following API request is used to create a generic secret with base64-encoded values, such as client credentials and other necessary fields, for integrating grounding in the generative AI hub. It can use OAuth2Password or OAuth2ClientCredentials to generate access tokens.

> **Note**
>
> In *API permissions*:
> - If you configure access from SAP BTP with **`OAuth2Password`** authentication, use the permission name **`Sites.Read.All`** with type **`Delegated`**
> - If you configure access from SAP BTP with **`OAuth2ClientCredentials`** authentication, use the permission name **`Sites.Selected`** with type **`Application`**
>
> For more information, see Integrating Joule with SAP Solutions: Configure Access from SAP BTP.

Examples

Example with OAuth2Password:

The `name` attribute is written without hyphens to make it simple to consume as a Unix environment variable later. It's written in capitals as is convention.

Field	Value
`name`	Name of the generic secret to be created
`description`	Base64 encoded value for the description of the generic secret to be created
`url`	Base64 encoded value for **`https://graph.microsoft.com`**
`authentication`	Base64 encoded value for **`OAuth2Password`**

Field	Value
`user`	Base64 encoded value for your Microsoft SharePoint technical user
`password`	Base64 encoded value for your Microsoft SharePoint password
`clientId`	Base64 encoded value for your Microsoft SharePoint Application credentials
`clientSecret`	Base64 encoded value for your Microsoft SharePoint Application credentials
`tokenServiceURL`	Base64 encoded value for `https://login.microsoftonline.com/<Microsoft EntraIDtenantidentifier>/oauth2/v2.0/token`

> **Note**
>
> The following attributes are set as default, and don't need to be passed during the API call:
>
> - `type`: `"HTTP"`
> - `proxyType`: `"Internet"`
> - `Scope`: `https://graph.microsoft.com/.default`

```
curl --location --request POST "$AI_API_URL/v2/admin/secrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <resource group created for grounding>' \
--data-raw '{
  "name": "<generic secret name>",
  "data": {
    "description": "<description of generic secret>",
    "clientId": "<client id>",
    "authentication": "T0F1dGgyUGFzc3dvcmQ=",
    "tokenServiceURL": "<token service url>",
    "password": "<password>",
    "url": "aHR0cHM6Ly9ncmFwaC5taWNyb3NvZnQuY29t",
    "user": "<user>",
    "clientSecret": "<client secret>",
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
     },
     {
      "key": "ext.ai.sap.com/documentRepositoryType",
       "value": "MSSharePoint"
    }
  ]
}'
```

Example with OAuth2ClientCredentials:

The `name` attribute is written without hyphens to make it simple to consume as a Unix environment variable later. It's written in capitals as is convention.

Field	Value
`name`	Name of the generic secret to be created
`description`	Base64 encoded value for the description of the generic secret to be created
`url`	Base64 encoded value for **`https://sap.sharepoint.com/sites/<site-name>`**
`authentication`	Base64 encoded value for **`OAuth2ClientCredentials`**
`clientId`	Base64 encoded value for your Microsoft SharePoint Application credentials
`clientSecret`	Base64 encoded value for your Microsoft SharePoint Application credentials
`tokenServiceURL`	Base64 encoded value for `https://login.microsoftonline.com/` **`<Microsoft EntraIDtenantidentifier>`**`/oauth2/v2.0/token`

> **Note**
>
> The following attributes are set as default, and don't need to be passed during the API call:
>
> - `type`: `"HTTP"`
> - `proxyType`: `"Internet"`
> - `tokenServiceURLType`: `"Dedicated"`
> - `Scope`: `https://graph.microsoft.com/.default`

> **Sample Code**

```
curl --request POST \
--url $AI_API_URL/v2/admin/secrets \
--header 'AI-Resource-Group: <resource group created for grounding>' \
--data-raw '{
  "name": "<generic secret name>",
  "data": {
    "description": "<description of generic secret>",
    "clientId": "<client id>",
    "authentication": "T0F1dGgyQ2xpZW50Q3JlZGVudGlhbHM=",
    "tokenServiceURL": "<token service
url>",
    "url":
"aHR0cHM6Ly9zYXAuc2hhcmVwb2ludC5jb20vc2l0ZXMvPHNpdGUtbmFtZT4=",

    "clientSecret": "<client secret>",
  },
   "labels": [
    {
```

```
      "key": "ext.ai.sap.com/document-grounding",
        "value": "true"
      },
      {
       "key": "ext.ai.sap.com/documentRepositoryType",
       "value": "MSSharePoint"
     }
   ]
}'
```

Next Steps

After registering your repository with grounding, you can use the document grounding APIs to prepare and store your vectors. For more information, see Pipelines API [page 438].

Related Information

Update a Generic Secret [page 715]
Delete a Generic Secret [page 718]

1.4.3.2.1.1.1.2.3 Grounding Generic Secrets for AWS S3

Your cloud storage credentials are managed using secrets. Secrets are a means of allowing and controlling connections across directories and tools, without compromising your credentials.

Context

You have created a resource group for grounding. For more information, see Create a Resource Group for Grounding [page 397].

Procedure

1. Send a POST request and enter the URL `{{apiurl}}/v2/admin/secrets`.
   Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN

Header	Value
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

The following API request is used to create a generic secret with base64-encoded values, such as client credentials and other necessary fields, for integrating grounding in the generative AI hub.

Example

The `name` attribute is written without hyphens to make it simple to consume as a Unix environment variable later. It is written in capitals as is convention.

Field	Value
`name`	Name of the generic secret to be created
`url`	Base64-encoded value in the format **`https://s3.<region>.amazonaws.com`**
`authentication`	Base64-encoded value for **`NoAuthentication`**
`description`	Base64-encoded value for the description of the generic secret to be created
`access_key_id`	Base64-encoded value for access key id
`bucket`	Base64-encoded value for AWS S3 bucket name
`host`	Base64-encoded value for AWS S3 host
`region`	Base64-encoded value for AWS S3 region
`secret_access_key`	Base64-encoded value for AWS S3 credentials
`username`	Base64-encoded value for AWS S3 credentials

> **Note**
>
> The following attributes are set as default and do not need to be passed during the API call:
>
> - `type`: `"HTTP"`
> - `proxyType`: `"Internet"`

```
curl --location --request POST "$AI_API_URL/v2/admin/secrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <resource group created for grounding>' \
--data-raw '{
```

```
  "name": "<generic secret name>",
  "data": {
    "url": "<url>",
    "authentication": "Tm9BdXRoZW50aWNhdGlvbg==",
    "description": "<description of generic secret>",
    "access_key_id": "<access key id>",
    "bucket": "<bucket>",
    "host": "<host>",
    "region": "<region>",
    "secret_access_key": "<secret access key>",
    "username": "<username>"
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
      "value": "S3"
    }
  ]
}'
```

Next Steps

After registering your repository with grounding, you can use the document grounding APIs to prepare and store your vectors. For more information, see Pipelines API [page 438].

Related Information

Update a Generic Secret [page 715]
Delete a Generic Secret [page 718]
Making requests using the REST API

1.4.3.2.1.1.1.2.4 Grounding Generic Secrets for SFTP

Your cloud storage credentials are managed using secrets. Secrets are a means of allowing and controlling connections across directories and tools, without compromising your credentials.

Prerequisites

You have created a resource group for grounding. For more information, see Create a Resource Group for Grounding [page 397].

Procedure

1. Send a POST request and enter the URL `{{apiurl}}/v2/admin/secrets`.
   Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

   The following API request is used to create a generic secret with base64-encoded values, such as client credentials and other necessary fields, for integrating grounding in the generative AI hub.

Example

The `name` attribute is written without hyphens to make it simple to consume as a Unix environment variable later. It is written in capitals as is convention.

Field	Value
`name`	Name of the generic secret to be created
`url`	Base64-encoded SFTP endpoint URL in the format **`https://<hostname>:<port>`**, where `<hostname>` and `<port>` represent the SFTP server host and port. Use this format when configuring the endpoint. For example, `https://sftp.example.com:22` **Note** Although the connection uses the SFTP protocol internally, the endpoint must be specified using the `https://` format **Restriction** Do not use `sftp://<hostname>:<port>`
`authentication`	Base64 encoded value for **`NoAuthentication`**
`description`	Base64 encoded value for the description of the generic secret to be created
`user`	Base64 encoded value for SFTP credentials

Field	Value
`password`	Base64 encoded value for SFTP credentials
`public_key`	The Base64-encoded SFTP server host public key is used to verify the server identity during connection setup and must be provided in addition to the SFTP user credentials.
`public_key_type`	Base64 encoded value for SFTP public key type: `ssh-ed25519` and `ssh-rsa`.

> **Note**
>
> The following attributes are set as default, and do not need to be passed during the API call:
>
> - `type`: `"HTTP"`
> - `proxyType`: `"Internet"`

Retrieve the SFTP Server Public Key

You can retrieve the SFTP server host public key using standard SSH utilities such as `ssh-keyscan`.

For more information, see https://man.openbsd.org/ssh-keyscan .

```
curl --location --request POST "$AI_API_URL/v2/admin/secrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <resource group created for grounding>' \
--data-raw '{
  "name": "<generic secret name>",
  "data": {
    "url": "<url>",
    "authentication": "Tm9BdXRoZW50aWNhdGlvbg==",
    "description": "<description of generic secret>",
    "password": "<password>",
    "user": "<user>",
    "public_key": "<public key>",
    "public_key_type": "<public key type>"
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
      "value": "SFTP"
    }
    }
  ]
}'
```

Next Steps

After registering your repository with grounding, you can use the document grounding APIs to prepare and store your vectors. For more information, see Pipelines API [page 438].

Related Information

Update a Generic Secret [page 715]
Delete a Generic Secret [page 718]

1.4.3.2.1.1.1.2.5 Grounding Generic Secrets for SAP Build Work Zone

Your cloud storage credentials are managed using secrets. Secrets are a means of allowing and controlling connections across directories and tools, without compromising your credentials.

Prerequisites

You've created a resource group for grounding. For more information, see Create a Resource Group for Grounding [page 397].

Procedure

1. Send a POST request and enter the URL `{{apiurl}}/v2/admin/secrets`.
   Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

   The following API request is used to create a generic secret with base64-encoded values, such as client credentials and other necessary fields, for integrating grounding in the generative AI hub. It can use OAuth2ClientCredentials with client secret.

Example

The `name` attribute is written without hyphens to make it simple to consume as a Unix environment variable later. It's written in capitals as is convention.

Field	Value
`name`	Name of the generic secret to be created
`description`	Base64 encoded value for the description of the generic secret to be created
`url`	Base64 encoded value for your SAP Build Work Zone URL
`authentication`	Base64 encoded value for **`OAuth2ClientCredentials`**
`clientId`	Base64 encoded value for client ID
`clientSecret`	Base64 encoded value for client secret
`tokenServiceURL`	Base64 encoded value for token service URL for client secret based authentication

> **Note**
>
> The following attributes are set as default, and don't need to be passed during the API call:
>
> - `type`: `"HTTP"`
> - `proxyType`: `"Internet"`
> - `tokenServiceURLType`: `"Dedicated"`

```
curl --request POST \
  --url $AI_API_URL/v2/admin/secrets \
  --header 'ai-resource-group: {{resource_group}}' \
  --header 'content-type: application/json' \
  --data '{
    "name": "<name-of-secret>",
    "data": {
    "description": "<any string base64 encoded>",
    "authentication": "T0F1dGgyQ2xpZW50Q3JlZGVudGlhbHM=",
    "url": "< base64 encoded SAP Build Work Zone-server-url>",
    "clientId": "< base64 encoded client-id>",
    "clientSecret": "< base64 encoded client-secret>",
    "tokenServiceURL": "< base64 encoded token-service-url>"
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
      "value": "WorkZone"
    }
  ]
}'
```

Next Steps

After registering your repository with grounding, you can use the document grounding APIs to prepare and store your vectors. For more information, see Pipelines API [page 438].

Related Information

Update a Generic Secret [page 715]
Delete a Generic Secret [page 718]
Integration With Document Grounding SAP Build Work Zone, Advanced Edition

1.4.3.2.1.1.1.2.6 Grounding Generic Secrets for SAP Document Management Service

Prerequisites

You have created a resource group for grounding. For more information, see Create a Resource Group for Grounding [page 397].

Procedure

1. Send a POST request and enter the URL `{{apiurl}}/v2/admin/secrets`.
   Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

   The following API request is used to create a generic secret with base64-encoded values, such as client credentials and other necessary fields, for integrating grounding in the generative AI hub. It can use OAuth2ClientCredentials with client secret.

Example

The `name` attribute is written without hyphens to make it simple to consume as a Unix environment variable later. It is written in capitals as is convention.

Field	Value
`name`	Name of the generic secret to be created
`description`	Base64 encoded value for the description of the generic secret to be created
`url`	Base64 encoded value for your AI Bridge Service URL and repository path. For example, `https://sdm-ai-bridge.cfapps.<region>.hana.ondemand.com/rest/v1/repositories/<repoID>/metadata`.
`authentication`	Base64 encoded value for **`OAuth2ClientCredentials`**
`clientId`	Base64 encoded value for client ID
`clientSecret`	Base64 encoded value for client secret
`tokenServiceURL`	Base64 encoded value for token service URL for client secret based authentication

> **Note**
>
> The following attributes are set as default, and do not need to be passed during the API call:
>
> - `type`: `"HTTP"`
> - `proxyType`: `"Internet"`
> - `tokenServiceURLType`: `"Dedicated"`

```
curl --request POST \
  --url $AI_API_URL/v2/admin/secrets \
  --header 'ai-resource-group: {{resource_group}}' \
  --header 'content-type: application/json' \
  --data '{
    "name": "<name-of-secret>",
    "data": {
    "description": "<any string base64 encoded>",
    "authentication": "T0F1dGgyQ2xpZW50Q3JlZGVudGlhbHM=",
    "url": "<base64 encoded AI Bridge Service URL>",
    "clientId": "< base64 encoded client-id>",
    "clientSecret": "< base64 encoded client-secret>",
    "tokenServiceURL": "< base64 encoded token-service-url>"
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
```

```
      "value": "SDM"
    }
  ]
}'
```

Next Steps

After registering your repository with grounding, you can use the document grounding APIs to prepare and store your vectors. For more information, see Pipelines API [page 438].

Related Information

Update a Generic Secret [page 715]
Delete a Generic Secret [page 718]
SAP Document Management Service
Access the Document Grounding using the SAP AI SDK for Java
Access the Document Grounding using the SAP AI SDK for JavaScript

1.4.3.2.1.1.1.2.7 Grounding Generic Secrets for ServiceNow

Your cloud storage credentials are managed using secrets. Secrets are a means of allowing and controlling connections across directories and tools, without compromising your credentials.

Prerequisites

You have created a resource group for grounding. For more information, see Create a Resource Group for Grounding [page 397].

Procedure

1. Send a POST request and enter the URL `{{apiurl}}/v2/admin/secrets`.
   Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN

Header	Value
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

The following API request is used to create a generic secret with base64-encoded values, such as client credentials and other necessary fields, for integrating grounding in the generative AI hub. It can use OAuth2ClientCredentials with client secret.

Example

The `name` attribute is written without hyphens to make it simple to consume as a Unix environment variable later. It is written in capitals as is convention.

Field	Value
`name`	Name of the generic secret to be created
`description`	Base64 encoded value for the description of the generic secret to be created
`url`	Base64 encoded value for your ServiceNow instance URL
`authentication`	Base64 encoded value for **`ClientCertificateAuthentication`**
`certificateType`	Base64 encoded value for pfx
`certificateContent`	Base64 encoded value for your certificate content
`certificatePassword`	Base64 encoded value for your certificate password

> **Note**
>
> The following attributes are set as default, and do not need to be passed during the API call:
>
> - `type`: `"HTTP"`
> - `proxyType`: `"Internet"`

```
 curl --location --request POST $AI_API_URL/v2/admin/secrets \
--header "Authorization: Bearer $TOKEN" \
--header "Content-Type: application/json" \
--header "AI-Resource-Group: <resource-group-for-grounding>" \
--data-raw ''{
  "name": "<GENERIC_SECRET_NAME>",
  "data": {
    "authentication": "Q2xpZW50Q2VydGlmaWNhdGVBdXRoZW50aWNhdGlvbg==",
    "description": "<BASE64_ENCODED_DESCRIPTION>",
    "certificateType": "cGZ4",
    "certificateContent": "<BASE64_ENCODED_CERT_CONTENT>",
```

```
    "certificatePassword": "<BASE64_ENCODED_CERT_PASSWORD>",
    "url": "<BASE64_ENCODED_SERVICENOW_URL>"
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
      "value": "ServiceNow"
    }
  ]
}'
```

Next Steps

After registering your repository with grounding, you can use the document grounding APIs to prepare and store your vectors. For more information, see Pipelines API [page 438].

Related Information

Update a Generic Secret [page 715]
Delete a Generic Secret [page 718]

1.4.3.2.1.1.1.3 Contextualization Using Metadata

1.4.3.2.1.1.1.3.1 Prepare your Metadata API Server

The metadata server enables grounding to process documents from your repositories, enriched with metadata for improved contextualization. To incorporate metadata into grounding, you must prepare, deploy and host a metadata server. The **default** path is `/v1/dg-pipeline/metadata` and the HTTP method is GET.

For metadata servers hosted at the default path, you only need to specify the **base** URL, for example, https://metadata-server.com in the **URL** field of the generic secret.

For metadata servers not hosted at the default path, you need to specify the **full** URL, for example, https://metadata-server.com/custom/path/to/metadata in the **URL** field of the generic secret.

Metadata API Response Structure

Your metadata server must return a response in the following format:

Output Code

```
{
  "documentsCount": 100,
  "value": [
    {
      "metadata": [
        {
          "key": "domainId",
          "value": ["PUBLIC", "PRIVATE"],
          "matchMode": "ANY"
        },
        {
          "key": "department",
          "value": ["HR", "Finance", "Engineering"],
          "matchMode": "ALL"
        }
      ],
      "documents": [
        {
          "id": "3893fe5e-9b22-4eb3-ad12-6a8b4b38f25h",
          "viewLocation": "/path/to/view/doc",
          "metadata": [
            {
              "key": "domainId",
              "value": ["PRIVATE"]
            },
            {
              "key": "department",
              "value": ["HR", "Finance"],
              "matchMode": "ALL"
            }
          ],
          "downloadLocation": "/path/to/download/doc",
          "lastUpdatedTimestamp": "2024-02-25T12:00:00Z"
        }
        // ... more documents
      ]
    }
    // ... more document groups
  ]
}
```

The top-level value is an array of document groups.

Each document group contains group-level `metadata` and a `documents` array.

Each document must have the following:

- A unique `id`
- `viewLocation`: the file location in the repository
- `lastUpdatedTimestamp`: the timestamp of the last update

Optionally, documents can also have the following;

- `metadata`: your choice of labels
- `downloadLocation`: the location from that the file can be downloaded from

The metadata keys and values are your own meaningful custom strings.

`matchMode` must be `ALL` to match all metadata values, or `ANY` to match any single value.

Authentication Requirements

Your metadata API must be secured using OAuth2ClientCredentials flow. You can use one of the following:

- Client secret-based authentication
- Mutual TLS (mTLS) with client certificates

For more information, see Generic Secrets for Contextualization with Metadata [page 419].

Ensure your server validates the incoming access token before serving the metadata.

Pagination

Pagination is optional.

If pagination is included, the following query parameters must be supported:

Parameter	Type	Description
`$top`	Integer	Max number of documents to return.
`$skip`	Integer	Number of documents to skip.
`$count`	Boolean	`true`: returns the total number of documents available on the server in the `documentsCount` field. `false`: returns the number of documents included in the current response in the `documentsCount` field.

Pagination requirements

Pagination must be applied across all document groups, meaning the API returns `$top`, `$count` and skip the first `$skip` documents globally, regardless of which group they belong to.

Document groups with no documents must be removed from the response.

The API must not return more documents than specified by `$top`.

The generic secret for the metadata server can include a `pageSize` attribute to set `$top`. For more information, see Generic Secrets for Contextualization with Metadata [page 419]

Example

A GET request to the endpoint `GET <BASE_URL>/v1/dg-pipeline/metadata?$top=2&$skip=1&$count=true` should return the following response:

Output Code

```
{
  "documentsCount": 5, // Total documents available on server (since
$count=true)
  "value": [
    {
      "metadata": [
```

```
        { "key": "domainId", "value": ["PUBLIC", "PRIVATE"], "matchMode":
"ANY" },
        { "key": "department", "value": ["HR", "Finance", "Engineering"],
"matchMode": "ALL" }
      ],
      "documents": [
        {
          "id": "doc-002-id",
          "viewLocation": "/path/to/view/doc2",
          "metadata": [
            { "key": "domainId", "value": ["PUBLIC"], "matchMode": "ANY" },
            { "key": "department", "value": ["HR", "Finance"], "matchMode":
"ALL" }
          ],
          "downloadLocation": "/path/to/download/doc2",
          "lastUpdatedTimestamp": "2024-02-26T12:00:00Z"
        },
        {
          "id": "doc-003-id",
          "viewLocation": "/path/to/view/doc3",
          "metadata": [
            { "key": "domainId", "value": ["PRIVATE"]}
          ],
          "downloadLocation": "/path/to/download/doc3",
          "lastUpdatedTimestamp": "2024-02-27T12:00:00Z"
        }
      ]
    }
  ]
}
```

Deployment

Now that you've prepared your application, you can deploy it. For more information, see SAP BTP Deploy an Application, DAP BTP Deploying to the Cloud Foundry Environment and Create an Application with Cloud Foundry Python Buildpack BTP Tutorial.

1.4.3.2.1.1.1.3.2 Generic Secrets for Contextualization with Metadata

Prerequisites

You have created a resource group for grounding. For more information, see Create a Resource Group for Grounding [page 397].

Procedure

1. Send a POST request and enter the URL `{{apiurl}}/v2/admin/secrets`.
   Include the following headers in your request:
   - `AI_API_URL`: the base URL of your SAP AI Core environment.
   - `{{access_token}}`: Your access token for SAP AI Core

   Choose one from the following headers to set your scope:
   - **`AI-Tenant-Scope`** : **`true`**. The operation is performed at the main tenant level.
   - **`AI-Resource-Group`** : **`<resource-group-for-grounding>`**. The operation is performed at the resource-group level.

   The following API request is used to create a generic secret with base64-encoded values, such as client credentials and other necessary fields, for integrating grounding in the generative AI hub. It can use OAuth2ClientCredentials with client secret or client certificate.

The `name` attribute is written without hyphens to make it simple to consume as a Unix environment variable later. It is written in capitals as is convention.

Common Fields

Field	Value
`name`	Name of the generic secret to be created
`description`	Base64 encoded value for the description of the generic secret to be created
`url`	Base64 encoded value for the metadata API server URL
`authentication`	Base64 encoded value for **`OAuth2ClientCredentials`**
`clientId`	Base64 encoded value for client ID
`pageSize` (optional)	If pagination is required, enter a Base64 encoded value of integer from 1 to 1000. **⚠ Restriction** Pagination and the page size attribute are only supported if you have a metadata server with pagination implemented. For more information, see Prepare your Metadata API Server [page 416].

With Client Secret

Field	Value
`clientSecret`	Base64 encoded value for client secret
`tokenServiceURL`	Base64 encoded value for token service URL for client secret based authentication

With Client Certificate

Field	Value
`certificateType`	Base64 encoded value for `P12` or `PEM`
`certificateContent`	Base64 encoded value for certificate content
`certificatePassword` (optional)	Base64 encoded value for password of the certificate if set
`tokenServiceURL`	Base64 encoded value for token service URL for certificate based authentication
`tokenServiceBodyParameters` (optional)	Base64 encoded value for valid JSON object with key-value pair

> **Note**
>
> The following attributes are set as default, and do not need to be passed during the API call:
>
> - `type`: `"HTTP"`
> - `proxyType`: `"Internet"`
> - `tokenServiceURLType`: `"Dedicated"`

Examples

Example with Client Secret:

> **Sample Code**

```
curl --location --request POST "$AI_API_URL/v2/admin/secrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <resource group created for grounding>' \
--data-raw '{
  "name": "<generic secret name>",
  "data": {
    "url": "<url>",
    "description": "<description of generic secret>",
    "authentication": "T0F1dGgyQ2xpZW50Q3JlZGVudGlhbHM=",
    "clientId": "<client id>",
    "clientSecret": "<client secret>",
    "tokenServiceURL": "<token service url for client secret based
authentication>",
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
      "value": "MetaData"
    }
  ]
```

```
}'
```

Example with Client Secret and Pagination:

Sample Code

```
curl --location --request POST "$AI_API_URL/v2/admin/secrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <resource group created for grounding>' \
--data-raw '{
  "name": "<generic secret name>",
  "data": {
    "url": "<url>",
    "description": "<description of generic secret>",
    "authentication": "T0F1dGgyQ2xpZW50Q3JlZGVudGlhbHM=",
    "clientId": "<client id>",
    "clientSecret": "<client secret>",
    "tokenServiceURL": "<token service url for client secret based
authentication>",
    "pageSize": "<Base64 encoded integer 1-1000>"
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
      "value": "MetaData"
    }
  ]
}'
```

Example with Client Certificate:

Sample Code

```
curl --location --request POST "$AI_API_URL/v2/admin/secrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <resource group created for grounding>' \
--data-raw '{
  "name": "<generic secret name>",
  "data": {
    "url": "<url>",
    "description": "<description of generic secret>",
    "authentication": "T0F1dGgyQ2xpZW50Q3JlZGVudGlhbHM=",
    "clientId": "<client id>",
    "certificateType": "UDEyCg== [P12] / UEVNCg== [PEM]",
    "certificateContent": "<certificate content>",
    "certificatePassword": "<certificate password>",
    "tokenServiceBodyParameters": "<valid json with key-value pairs>",
    "tokenServiceURL": "<token service url for certificate based
authentication>
```

```
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
      "value": "MetaData"
    }
  ]
}'
```

Example with Client Certificate and Pagination:

Sample Code

```
curl --location --request POST "$AI_API_URL/v2/admin/secrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <resource group created for grounding>' \
--data-raw '{
  "name": "<generic secret name>",
  "data": {
    "url": "<url>",
    "description": "<description of generic secret>",
    "authentication": "T0F1dGgyQ2xpZW50Q3JlZGVudGlhbHM=",
    "clientId": "<client id>",
    "certificateType": "UDEyCg== [P12] / UEVNCg== [PEM]",
    "certificateContent": "<certificate content>",
    "certificatePassword": "<certificate password>",
    "tokenServiceBodyParameters": "<valid json with key-value pairs>",
    "tokenServiceURL": "<token service url for certificate based
authentication>"
    "pageSize": "<integer 1-1000>"
  },
  "labels": [
    {
      "key": "ext.ai.sap.com/document-grounding",
      "value": "true"
    },
    {
      "key": "ext.ai.sap.com/documentRepositoryType",
      "value": "MetaData"
    }
  ]
}'
```

1.4.3.2.1.1.2 Data Management APIs

Using the Document Grounding APIs, you can automate the process of retrieving, preprocessing, and embedding documents from supported data sources into the SAP HANA Vector Store. These APIs handle the end-to-end flow of documents, including document fetching, text chunking, and embedding via the Vector API, without requiring you to manage each step manually.

For document processing and vector generation, the following options are available:

- For use cases where documents are stored in a repository, the **Pipelines API** is recommended.
  The **Pipelines API** creates data management pipelines that fetch documents from a supported data source, preprocess and chunk those documents, and store their semantic embeddings in the HANA Vector Store. For more information, see Pipelines API [page 438].
- For use cases where documents are uploaded and managed directly, the **Vector API** is recommended.
  The **Vector API** preprocesses documents into chunks and stores their semantic embeddings in collections. For more information, see Vector API [page 477].

The **Retrieval API** performs similarity searches on the vector database for information retrieval. It can be used for repositories of documents that have been processed using the Pipelines API, or for collections that have been processed using the Vector API. For more information, see Preparing Data Using the Vector API [page 479].

> → Tip
>
> If you use the Pipelines API, you don't also need to call the Vector API. After the data is embedded, you can directly use the Retrieval API to query the vector store for relevant results.

For more information about the API, such as request formats, response schemas, authentication headers, and supported operations, see Grounding APIs in the Business Accelerator Hub.

Rate Limits

Tenant Level

The following rate limits apply:

Operation Type	Call Type	Rate Limit
write_pipeline	POST – Create pipeline	5 requests per minute
write	PUT, POST, DELETE, PATCH	6 requests per second
read	GET	1200 requests per minute

Resource Group Level

The following rate limits apply:

Pipelines

Operation Type	Call Type	Rate Limit
write_pipeline	POST – Create pipeline	5 requests per minute
write	PUT, POST, DELETE, PATCH	6 requests per second
read	GET	1200 requests per minute

Metadata Management

Operation Type	Call Type	Rate Limit
write_pipeline	POST – Create pipeline	5 requests per minute
write	PUT, POST, DELETE, PATCH	6 requests per second
read	GET	1200 requests per minute

Vector

Endpoint	Call Type	Rate Limit
`$AI_API_URL/v2/lm/document-grounding/vector/search`	POST	250 requests per minute
`$AI_API_URL/v2/lm/document-grounding/vector/documents`	DELETE	
`$AI_API_URL/v2/lm/document-grounding/vector/documents/metadata`	PATCH	
`$AI_API_URL/v2/lm/document-grounding/vector/chunks/metadata`	PATCH	
`$AI_API_URL/v2/lm/document-grounding/vector/collections`	GET	
`$AI_API_URL/v2/lm/document-grounding/vector/collections`	POST	
`$AI_API_URL/v2/lm/document-grounding/vector/collections/metadata`	PATCH	
`$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}`	DELETE	
`$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents`	GET	
`$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents`	POST	

Endpoint	Call Type	Rate Limit
`$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents`	PATCH	
`$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents/{{documentId}}`	GET	
`$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents/{{documentId}}`	DELETE	
`$AI_API_URL/v2/lm/document-grounding/vector/collections/{{id}}/creationStatus`	GET	500 requests per minute
`$AI_API_URL/v2/lm/document-grounding/vector/collections/{{id}}/deletionStatus`	GET	
`$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}`	GET	

Retrieval

Endpoint	Call Type	Rate Limit
`$AI_API_URL/v2/lm/document-grounding/retrieval/dataRepositories`	GET	250 requests per minute
`$AI_API_URL/v2/lm/document-grounding/retrieval/search`	POST	
`$AI_API_URL/v2/lm/document-grounding/retrieval/dataRepositories/{{repositoryId}}`	GET	500 requests per minute

Capacity Quotas and Input Limits

Vector Ingestion – Capacity Limits

The following capacity limits apply:

Category	Description	Limit Type	Limit
Collections	Max collections	Quota	55,000
Documents	Max documents per collection	Quota	100,000
Documents	Max documents per ingestion request	Input Limit	100
Chunks	Max chunks per document	Quota	200,000

Metadata Limits (Ingestion and Query Time)

The following metadata limits apply:

Metadata Type	Limit Type	Limit
Collection Metadata	Quota / Input Limit	500 / 5,000
Document Metadata	Quota / Input Limit	500 / 5,000
Chunk Metadata	Quota / Input Limit	500 / 5,000

Vector Search – Query Limits

The following query limits apply:

Category	Description	Limit Type	Limit
Filters	Max filters per query	Input Limit	15
Collections	Max collection IDs per query	Input Limit	5,000
Results	Max chunk results per query	Input Limit	100
Results	Max document results per query	Input Limit	10

Retrieval Search – Query Limits

The following query limits apply:

Category	Description	Limit Type	Limit
Filters	Max filters per query	Input Limit	15
Collections	Max collection IDs per query	Input Limit	5,000
Results	Max chunk results per query	Input Limit	100
Results	Max document results per query	Input Limit	10

1.4.3.2.1.1.2.1 Metadata Management API

Overview

The Metadata Management API provides endpoints for managing metadata configurations and document metadata within the document grounding pipelines. You can create and manage metadata configurations for various data repositories, enumerate documents, and update document metadata in batch operations.

Key Capabilities

The Metadata Management API enables you to:

- Create and manage metadata configurations for different repository types.
- Retrieve metadata configurations with pagination support and detailed status information.
- Enumerate documents automatically from configured repositories.
- List and filter documents within metadata configurations using path-based queries.
- Update document metadata in batch for up to 1,000 documents per request.
- Manage the lifecycle of your metadata configurations.

API Request Format

When you build your API requests, follow the format outlined in the *Pipelines* section of the Grounding API. For more information, see Grounding API.

The endpoint for the pipelines API is `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations`.

1.4.3.2.1.1.2.1.1 Create a Metadata Configuration

Creates a new metadata configuration and triggers document enumeration for the configured repository.

Context

You can create a metadata configuration for reuse across your pipelines, instead of defining a metadata configuration for each pipeline.

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- You have created a generic secret. For more information, see Generic Secrets for Grounding [page 398].
- You've added your grounding documents to your document store and you're using a supported data source and document type. For more information, see Pipelines API [page 438].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations`.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Populate the sample code with the following values:

Field	Value
name	A unique identifier for your configuration
destinationName	The name of the generic secret for your data repository
dataRepositoryType	Your data repository type. Choose from: • `MSSharePoint` • `S3` • `SFTP`
includePaths	The paths to the documents to be included, in a comma separated list.
labels	Your metadata labels in key-value pairs.

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/metadata/
configurations \
  --header 'AI-Resource-Group: <resource_group>' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer $TOKEN' \
  --data '{
      "name": "config-name",
      "destinationName": "destination-name",
      "dataRepositoryType": "<data repository type>",
```

```
      "includePaths": [
        "/path/one",
        "/path/two"
      ],
      "labels": [
        {
          "key": "department",
          "value": "finance"
        }
      ]
    }'
```

Results

Successful responses return code **202** and include a success message.

The response includes details of your metadata configuration resource, including the **metadata configuration ID**.

Document enumeration is performed asynchronously after the request is accepted.

Next Steps

You can reference your metadata configuation as part of a pipeline by referencing the metadata configuration ID when you create your pipeline. For more information, see Create a Pipeline using a Metadata Configuration [page 462].

1.4.3.2.1.1.2.1.2 List Metadata Configurations

Retrieves a paginated list of all available metadata configurations.

Procedure

1. Send a GET request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations`.
   You can include optional parameters for pagination support. The following query parameters are supported:

Parameter	Type	Description
`$top`	Integer	Max number of documents to return.

Parameter	Type	Description
`$skip`	Integer	Number of documents to skip.
`$count`	Boolean	`true`: returns the total number of documents available on the server in the `documentsCount` field. `false`: returns the number of documents included in the current response in the `documentsCount` field.

For example: `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations?top=10&skip=0`

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/metadata/
configurations \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

Results

Successful responses return code **200** and include details of your **metadata configurations** in JSON format.

Output Code

```
{
  "count": 1,
  "resources": [
    {
      "id": "uuid",
      "name": "Config Name",
      "destinationName": "destination",
      "dataRepositoryType": "MSSharePoint",
      "includePaths": [
        "/path"
      ],
      "enumerationStatus": "IN_PROGRESS"
```

```
    }
  ]
}
```

Enumeration status values include:

- NEW
- IN_PROGRESS
- COMPLETE
- ERROR

1.4.3.2.1.1.2.1.3 Get a Metadata Configuration by ID

Retrieves the details of a specific metadata configuration by its ID.

Procedure

1. Send a GET request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations/<metadataConfigId>`.
   Include the ID of the metadata configuration that you want to retrieve in your path.
   Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/metadata/
configurations/<metadataConfigId> \
  --header 'AI-Resource-Group: <resource_group>' \
  --header 'Authorization: Bearer $TOKEN'
```

Results

Successful responses return code **200** and include details of your metadata configuration in JSON format.

Error responses include **404**: Metadata configuration not found

1.4.3.2.1.1.2.1.4 Delete a Metadata Configuration

Delete a specific metadata configuration by its ID.

Prerequisites

You've deleted any pipelines that are dependent on the metadata configuration that you want to delete.

Procedure

1. Send a DELETE request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations/<metadataConfigId>`.
   Include the ID of the metadata configuration that you want to retrieve in your path.
   Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Sample Code

```
curl --request DELETE \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/metadata/
configurations/<metadataConfigId> \
  --header 'AI-Resource-Group: <resource_group>' \
  --header 'Authorization: Bearer $TOKEN'
```

Results

Successful responses return code **202** and include a success message.

Error responses include:

- **404**: Metadata configuration not found
- **409**: Pipelines exist for this configuation
- **500**: Internal server error

1.4.3.2.1.1.2.1.5 List the Documents of a Metadata Configuration

Retrieves a paginated list of documents associated with a metadata configuration.

Procedure

1. Send a GET request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations/<metadataConfigId>/documents`.
   You can include optional parameters for pagination support. The following query parameters are supported:

Parameter	Type	Description
`absolutePath`	String	The file path to restrict the search to. Wildcards are supported.
`$top`	Integer	Max number of documents to return.
`$skip`	Integer	Number of documents to skip.
`$count`	Boolean	`true`: returns the total number of documents available on the server in the `documentsCount` field. `false`: returns the number of documents included in the current response in the `documentsCount` field.

   For example: `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations/<metadataConfigId>/documents?top=10&skip=0`

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/metadata/
configurations/<metadataConfigId>/documents \
  --header 'AI-Resource-Group: <resource_group>' \
```

```
  --header 'Authorization: Bearer $TOKEN'
```

Results

Successful responses return code **200** and include details of the documents associated with your metadata configurations in JSON format.

Output Code

```
{
  "count": 1,
  "resources": [
    {
      "id": "doc-uuid",
      "title": "Document Title",
      "absoluteFilePath": "/folder/file.pdf",
      "createdTimestamp": "2025-08-28T06:15:30Z",
      "type": "DOCUMENT",
      "metadata": []
    }
  ]
}
```

Type values include:

- FOLDER
- DOCUMENT

1.4.3.2.1.1.2.1.6 Get Details of a Document

Retrieves details of a document associated with a metadata configuration by document ID.

Procedure

1. Send a GET request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations/<metadataConfigId>/documents/<documentId>`.
   Include your metadata configuration ID and document ID in your request.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps

Header	Value
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/metadata/
configurations/<metadataConfigId>/documents/<documentId> \
  --header 'AI-Resource-Group: <resource_group>' \
  --header 'Authorization: Bearer $TOKEN'
```

Results

Successful responses return code **200** and include details of the document in JSON format.

Output Code

```
{
  "id": "doc-uuid",
  "title": "Document Title",
  "absoluteFilePath": "/folder/file.pdf",
  "createdTimestamp": "2025-08-28T06:15:30Z",
  "mimeType": "application/pdf",
  "fileSizeMb": "1.5",
  "metadata": []
}
```

Error responses include **404**:Document not found

1.4.3.2.1.1.2.1.7 Batch Update Document Metadata

Updates the metadata for multiple documents within a specific metadata configuration in a single request.

Context

The following limits apply:

- Maximum 1000 documents per request
- Maximum 10 metadata entries per document

Procedure

1. Send a PATCH request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/metadata/configurations/<metadataConfigId>/documents`.
   Include your metadata configuration ID in your request.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Populate the sample code with the following values:

Field	Value
id	The ID of the document to be updated
labels	Your metadata labels in key-value pairs.

matchMode	One of the following modes:
	• ANY
	  • At least one metadata value matches.
	  • This is the default match mode.
	  • Defined only at injection time.
	• ALL
	  • All document metadata values must match exactly, or be a subset of the search metadata.
	  • Supported only at the document level.
	  • Defined only at injection time.

Sample Code

```
curl --request PATCH \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/metadata/
configurations/<metadataConfigId>/documents/<documentId> \
  --header 'AI-Resource-Group: <resource_group>' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer $TOKEN' \
  --data '{
  "value": [
    {
      "id": "<documentId>",
      "metadata": [
        {
          "key": "contentObjectId",
          "value": [
            "12345"
          ],
          "matchMode": "ANY"
        }
      ]
```

```
    }
  ]
}'
```

Results

Successful responses return code **200** and include details of the document in JSON format.

Output Code

```
{
"id": "document-uuid",
"metadata": [...]
}
```

1.4.3.2.1.1.2.2 Pipelines API

The Pipelines API allows you to provide unstructured documents from your data repositories, and create a document grounding pipeline for your resource group.

The API is proxied through the generative AI hub and incorporates vector stores, such as the managed HANA database.

Each pipeline represents a configured end-to-end process including the following steps:

1. Fetches documents from a supported data source.
2. Preprocesses and chunks the document content, and generates semantic embeddings. Semantic embeddings are multidimensional representations of textual information.
3. Stores semantic embeddings into the HANA Vector Store

Data Repositories

The following repositories and document types are supported:

- Microsoft SharePoint
- AWS S3
- SFTP
- SAP Build Work Zone
- SAP Document Management service
- Service Now
- Google Drive

Document Repository	Document (File) Format	Content Refresh	Document Content	Maximum Number of Documents
Microsoft SharePoint	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT, JSON	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline
AWS S3	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT, JSON	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline
SFTP	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline
SAP Build Work Zone	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline
SAP Document Management service	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline
Service Now	Knowledge Base Articles in HTML format	Daily	Plain Text	Maximum of 8000 Documents per Pipeline
Google Drive	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT,JSON	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 8000 Documents per Pipeline

API Request Format

When you build your API requests, follow the format outlined in the *Pipelines* section of the Grounding API. For more information, see Grounding API.

The endpoint for the pipelines API is `$AI_API_URL/v2/lm/document-grounding/pipelines`.

For region and pricing information, see SAP Note 3437766.

> ⚠ Caution
>
> Do not expose personally identifiable or privileged information in the documents exposed to your pipeline when using generative AI hub. Personally identifiable information is any data that can be used alone, or in combination, to identify the person that the data refers to. The document grounding capability has no mechanism for determining the type of data that it processes, and does not filter confidential or other privileged information.

Related Information

SAP Document Management Service

1.4.3.2.1.1.2.2.1 Create a Document Grounding Pipeline Using the Pipelines API

This API call creates a pipeline for indexing documents for a resource group.

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- You have created a generic secret. For more information, see Generic Secrets for Grounding [page 398].
- You've added your grounding documents to your document store and you're using a supported data source and document type. For more information, see Pipelines API [page 438].
- If you're using metadata, you've configured a metadata server. For more information, see Prepare your Metadata API Server [page 416].

Context

You can schedule automatic content updates by using a `cronExpression` when you create your pipeline. For more information, see Cron Expressions [page 463].

For repositories that support it, you can include metadata in your pipeline. Metadata is additional information associated with data that can help in identifying, categorizing, and retrieving relevant content.

> → Tip
>
> If you use the pipelines API, you do not need to call the Vector API separately. After the data is embedded, you can directly use the Retrieval API to query the vector store for relevant sections.

Procedure

You document grounding pipeline with your chosen repository. For more information, see the following:

- Create a Pipeline with Microsoft SharePoint [page 273]
- Create a Pipeline with AWS S3 [page 277]
- Create a Pipeline with SFTP [page 280]

- Create a Pipeline with SAP Build Work Zone [page 284]
- Create a Pipeline with SAP Document Management Service [page 458]
- Create a Document Grounding Pipeline Using the Pipelines API [page 440]

Next Steps

You can use API endpoints to view and manage the lifecycle of your data pipelines, executions and documents. For more information, see Data Pipelines [page 465], Executions [page 469] and Documents [page 471].

You can manually restart a pipeline. For more information, see Manually Restart a Document Grounding Pipeline [page 475].

1.4.3.2.1.1.2.2.1.1 Create a Pipeline with Google Drive

Creates a document grounding pipeline that indexes content from Google Drive using a generic secret, with optional metadata and scheduled updates.

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- The generic secret for Google Drive is available. For more information, see Create a Generic Secret for Google Drive [page 227].
- You have created a generic secret for Google Drive. For more information, see Generic Secrets for Grounding [page 398].
- You have added your grounding documents to Google Drive using a supported document type. For more information, see Pipelines API [page 438].
- If you are using metadata, you have configured a metadata server. For more information, see Prepare your Metadata API Server [page 416].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines`.

Without Metadata

The following shows an example of a Google Drive resource to be indexed.

Populate the sample code with the following values:

Field	Value
`{{resource_group}}`	The AI resource group assigned to your account
`$AI_API_URL`	The base URL of your SAP AI Core environment. This can also be set as an environment variable
`{{access_token}}`	Your access token for SAP AI Core
`<generic secret name>`	The name of the generic secret created for Google Drive
`<resource id>`	The ID of the shared drive or shared folder

Sample Code

```
curl --request POST \
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines" \
  --header "AI-Resource-Group: <resource-group>" \
  --header "Content-Type: application/json" \
  --header "Authorization: Bearer $TOKEN" \
  --data '{
    "type": "GoogleDrive",
    "configuration": {
      "destination": "<GENERIC_SECRET_NAME>",
      "googleDrive": {
        "resourceType": "<SHARED_DRIVE_OR_SHARED_FOLDER>",
        "resourceId": "<SHARED_DRIVE_ID_OR_SHARED_FOLDER_ID>",
        "includePaths": [
          "path/to/folder1",
          "path/to/folder2"
        ]
      }
    }
  }'
```

With Metadata

Metadata is additional information associated with data that can help in identifying, categorizing, and retrieving relevant content.

→ Tip

Alternatively, you can create a reusable metadata configurations and then reference your configuration when creating your pipeline. For more information, see Create a Metadata Configuration [page 428] and Create a Pipeline using a Metadata Configuration [page 462].

The metadata attribute is optional and must be used only if a metadata server is configured. To create a pipeline without metadata, see Create a Document Grounding Pipeline Using the Pipelines API [page 440].

Sample Code

```
curl --request POST \
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines" \
```

```
  --header "AI-Resource-Group: <resource-group>" \
  --header "Content-Type: application/json" \
  --header "Authorization: Bearer $TOKEN" \
  --data '{
    "type": "GoogleDrive",
    "configuration": {
      "destination": "<GENERIC_SECRET_NAME>",
      "googleDrive": {
        "resourceType": "<SHARED_DRIVE_OR_SHARED_FOLDER>",
        "resourceId": "<SHARED_DRIVE_ID_OR_SHARED_FOLDER_ID>"
      }
    },
    "metadata": {
      "destination": "<METADATA_SECRET_NAME>"
    }
  }'
```

With Scheduled Updates

`cronExpression` is optional. You can add it to your `configuration` to schedule pipeline updates at intervals of your choice.

Sample Code

```
curl --request POST \
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines" \
  --header "AI-Resource-Group: <resource-group>" \
  --header "Content-Type: application/json" \
  --header "Authorization: Bearer $TOKEN" \
  --data '{
    "type": "GoogleDrive",
    "configuration": {
      "destination": "<GEN>",
      "googleDrive": {
        "resourceType": "<SHARED_DRIVE_OR_SHARED_FOLDER>",
        "resourceId": "<SHARED_DRIVE_ID_OR_SHARED_FOLDER_ID>"
      },
      "cronExpression": "<CRON_EXPRESSION>"
    }
  }'
```

Pipeline Configuration Options

Google Drive pipelines support two resource types and an optional auto-discovery mode.

Parameters

- `resourceType` : Specifies the Google Drive resource type. Valid values are `SHARED_FOLDER` and `SHARED_DRIVE`.
- `resourceId`: Specifies the unique ID of the Google Drive folder or shared drive. If this parameter is omitted, auto-discovery mode is enabled and the pipeline scans all resources shared with the service account.

- `includePaths`: Specifies an array of strings to filter specific sub-folders. Example: `["ProjectDocs/Final"]`.

> **Note**
>
> If `resourceId` is not specified, the pipeline runs in auto-discovery mode and retrieves all files that the service account has permission to read.

Next Steps

After preparing your vectors, you can use the Retrieval API for chunks relevant to a query, or use the grounding module as part of an orchestration workflow for information retrieval and LLM interaction. For more information, see Retrieval API [page 504] or Using the Grounding Module [page 510].

You can add metadata to your pipeline, or update existing metadata by sending a patch request. For more information, see Update Metadata [page 464].

Supported File Types

The Google Drive pipeline supports common document and image file types. Google Workspace native formats are handled by remapping supported files to equivalent Microsoft Office formats. Unsupported file types are skipped during pipeline execution.

Supported file types:

- DOCX
- HTML (static)
- JPEG
- JPG
- JSON
- PDF
- PNG
- PPTX
- TIFF
- TXT

Google Drive-specific file types:

- Google Docs (`.gdoc`) are remapped to `.docx`.
- Google Slides (`.gslides`) are remapped to `.pptx`.

Unsupported file types:

- Other Google Workspace formats, such as Google Sheets
- Other file formats, such as `.xlsx`

1.4.3.2.1.1.2.2.1.2 Create a Pipeline with Microsoft SharePoint

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- You have created a generic secret. For more information, see Generic Secrets for Grounding [page 398].
- You've added your grounding documents to your document store and you're using a supported data source and document type. For more information, see Pipelines API [page 438].
- If you're using metadata, you've configured a metadata server. For more information, see Prepare your Metadata API Server [page 416].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines`.

Without Metadata

The following shows an example of a SharePoint site to be indexed.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name>	The name of the generic secret created for MS SharePoint
<site name>	The name of the SharePoint site to be indexed
["/<folder1>", "/<folder2>/<sub-folder1>"]	An array of folders within the SharePoint site for selective indexing (Optional)

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
```

```
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "MSSharePoint",
  "configuration": {
    "destination": "<generic secret name>",
    "sharePoint": {
      "site": {
        "name": "<site name>",
        "includePaths": ["/<folder1>", "/<folder2>/<sub-folder1>"]
      }
    }
  }
}'
```

> **Note**
>
> The `includePaths` parameter is optional. It takes an array of folder paths within the SharePoint site to be indexed and limits the scope of the pipeline to the specified sub folders or files, meaning that only this content is indexed.
>
> The `includePaths` parameter input is restricted to folder paths within the default document folder of a SharePoint site. No other drives, document library paths, or formats are supported.

With Metadata

Metadata is additional information associated with data that can help in identifying, categorizing, and retrieving relevant content.

> **→ Tip**
>
> Alternatively, you can create a reusable metadata configurations and then reference your configuration when creating your pipeline. For more information, see Create a Metadata Configuration [page 428] and Create a Pipeline using a Metadata Configuration [page 462].

The metadata attribute is optional. It accepts the destination name, which is used to connect to the Microsoft SharePoint metadata server to retrieve metadata for document indexing.

The metadata attribute should be used only if a metadata server is configured. To create a grounding pipeline without metadata, see Create a Document Grounding Pipeline Using the Pipelines API [page 440].

The following shows an example of a SharePoint site to be indexed with metadata.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core

Field	Value
<generic secret name> (destination)	The name of the generic secret created for Microsoft SharePoint
<site name>	The name of the SharePoint site to be indexed
<generic secret name> (metadata)	The name of the generic secret created for Microsoft SharePoint Metadata Server

⚠ Restriction

`metadata` and `includePaths` cannot be used together in a single payload request.

{·} Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "MSSharePoint",
  "configuration": {
    "destination": "<generic secret name>",
    "sharePoint": {
      "site": {
        "name": "<site name>"
      }
    }
  },
  "metadata": {
    "destination": "<generic secret name>"
  }
}'
```

You can search a pipeline later in your workflow by using the `dataRepositoryMetadata` attribute in the `metadata` field. Metadata organizes information using categories (keys) paired with their related values. For example:

```
...
    "metadata": {
        "dataRepositoryMetadata": [
            {
                "key": "example",
                "value": [
                    "demo"
                ]
            }
        ]
    }
...
```

For more information, see Search a Pipeline [page 476].

With Scheduled Updates

`cronExpression` is optional. You can add it to your `configuration` to schedule updates to your pipeline, at intervals of your choice.

The following shows an example of a SharePoint site to be indexed at regular intervals.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name>	The name of the generic secret created for MS SharePoint
<site name>	The name of the SharePoint site to be indexed
cronExpression	The cronExpression for the update frequency of your choice (optional). For more information, see Cron Expressions [page 463].

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "MSSharePoint",
  "configuration": {
    "destination": "<generic secret name>",
    "cronExpression":"<cron expression>"
    "sharePoint": {
      "site": {
        "name": "<site name>"
      }
    },
  }
}'
```

Note

Each repetition of your pipeline is a new deployment, and incurs costs.

Next Steps

After preparing your vectors, you can use the Retrieval API for chunks relevant to a query, or use the grounding module as part of an orchestration workflow for information retrieval and LLM interaction. For more information, see Retrieval API [page 504] or Using the Grounding Module [page 510].

You can add metadata to your pipeline, or update existing metadata by sending a patch request. For more information, see Update Metadata [page 464].

1.4.3.2.1.1.2.2.1.3 Create a Pipeline with AWS S3

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- You have created a generic secret. For more information, see Generic Secrets for Grounding [page 398].
- You've added your grounding documents to your document store and you're using a supported data source and document type. For more information, see Pipelines API [page 438].
- If you're using metadata, you've configured a metadata server. For more information, see Prepare your Metadata API Server [page 416].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines`.

Without Metadata

The following shows an example of an AWS S3 repository to be indexed.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	{{access_token}}
<generic secret name>	Name of the generic secret created for AWS S3

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
```

```
  --data '{
  "type": "S3",
  "configuration": {
    "destination": "<generic secret name>",
    "s3": {
        "includePaths": ["/Sample Files/folder1"]
    }
  }
}'
```

With Metadata

Metadata is additional information associated with data that can help in identifying, categorizing, and retrieving relevant content.

> → Tip
>
> Alternatively, you can create a reusable metadata configurations and then reference your configuration when creating your pipeline. For more information, see Create a Metadata Configuration [page 428] and Create a Pipeline using a Metadata Configuration [page 462].

The metadata attribute is optional. It accepts the destination name, which is used to connect to the Microsoft SharePoint metadata server to retrieve metadata for document indexing.

The metadata attribute should be used only if a metadata server is configured. To create a grounding pipeline without metadata, see Create a Document Grounding Pipeline Using the Pipelines API [page 440].

The following shows an example of an AWS S3 repository to be indexed with metadata:

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name> (configuation)	The name of the generic secret created for AWS S3
<generic secret name> (metadata)	The name of the generic secret created for the AWS S3 Metadata Server

> ⚠ Restriction
>
> `metadata` and `includePaths` cannot be used together in a single payload request.

> {·} Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
```

```
   --data '{
   "type": "S3",
   "configuration": {
    "destination": "<generic secret name>",
     "s3": {
         "includePaths": ["/Sample Files/folder1"]
     }
   },
   "metadata": {
     "destination": "<generic secret name>"
   }
}'
```

You can search a pipeline later in your workflow by using the `dataRepositoryMetadata` attribute in the `metadata` field. Metadata organizes information using categories (keys) paired with their related values. For example:

```
...
    "metadata": {
        "dataRepositoryMetadata": [
            {
                "key": "example",
                "value": [
                    "demo"
                ]
            }
        ]
    }
...
```

For more information, see Search a Pipeline [page 476].

With Scheduled Updates

`cronExpression` is optional. You can add it to your `configuration` to schedule updates to your pipeline, at intervals of your choice.

The following shows an example of an AWS S3 repository to be indexed at regular intervals.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name>	Name of the generic secret created for AWS S3
cronExpression	The cronExpression for the update frequency of your choice (optional). For more information, see Cron Expressions [page 463].

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
    "type": "S3",
    "configuration": {
        "destination": "<generic secret name>",
        "cronExpression": "<cron expression>"
        "s3": {
            "includePaths": [
                "/Sample Files/folder1"
            ],
        }
    }
}'
```

Note

Each repetition of your pipeline is a new deployment, and incurs costs.

Next Steps

After preparing your vectors, you can use the Retrieval API for chunks relevant to a query, or use the grounding module as part of an orchestration workflow for information retrieval and LLM interaction. For more information, see Retrieval API [page 504] or Using the Grounding Module [page 510].

You can add metadata to your pipeline, or update existing metadata by sending a patch request. For more information, see Update Metadata [page 464].

1.4.3.2.1.1.2.2.1.4 Create a Pipeline with SFTP

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- You have created a generic secret. For more information, see Generic Secrets for Grounding [page 398].
- You've added your grounding documents to your document store and you're using a supported data source and document type. For more information, see Pipelines API [page 438].
- If you're using metadata, you've configured a metadata server. For more information, see Prepare your Metadata API Server [page 416].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines`.

Without Metadata

The following shows an example of an SFTP repository to be indexed:

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name>	The name of the generic secret created for SFTP

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "SFTP",
  "configuration": {
    "destination": "<generic secret name>",
    "sftp": {
        "includePaths": ["/Sample Files/folder1"]
    }
  }
}'
```

With Metadata

Metadata is additional information associated with data that can help in identifying, categorizing, and retrieving relevant content.

→ Tip

Alternatively, you can create a reusable metadata configurations and then reference your configuration when creating your pipeline. For more information, see Create a Metadata Configuration [page 428] and Create a Pipeline using a Metadata Configuration [page 462].

The metadata attribute is optional. It accepts the destination name, which is used to connect to the Microsoft SharePoint metadata server to retrieve metadata for document indexing.

The metadata attribute should be used only if a metadata server is configured. To create a grounding pipeline without metadata, see Create a Document Grounding Pipeline Using the Pipelines API [page 440].

The following shows an example of an SFTP repository to be indexed with metadata:

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name> (configuation)	The name of the generic secret created for SFTP
<generic secret name> (metadata)	The name of the generic secret created for the SFTP Metadata Server

⚠ Restriction

`metadata` and `includePaths` cannot be used together in a single payload request.

{·} Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "SFTP",
  "configuration": {
    "destination": "<generic secret name>",
    "sftp": {
        "includePaths": ["/Sample Files/folder1"]
    },
  "metadata": {
    "destination": "<generic secret name>"
  }
}'
```

The metadata attribute is optional. It accepts the destination name, which is used to connect to the SFTP metadata server to retrieve metadata for document indexing.

You can search a pipeline later in your workflow by using the `dataRepositoryMetadata` attribute in the `metadata` field. Metadata organizes information using categories (keys) paired with their related values. For example:

```
...
    "metadata": {
        "dataRepositoryMetadata": [
            {
                "key": "example",
                "value": [
                    "demo"
                ]
            }
        ]
```

```
    }
...
```

For more information, see Search a Pipeline [page 476].

With Scheduled Updates

`cronExpression` is optional. You can add it to your `configuration` to schedule updates to your pipeline, at intervals of your choice.

The following shows an example of an SFTP repository to be indexed at regular intervals.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name>	The name of the generic secret created for SFTP
cronExpression	The cronExpression for the update frequency of your choice (optional). For more information, see Cron Expressions [page 463].

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
    "type": "SFTP",
    "configuration": {
        "destination": "<generic secret name>",
        "cronExpression": "<cron expression>",
        "sftp": {
            "includePaths": [
                "/Sample Files/folder1"
            ]
        }
    }
}'
```

Note

Each repetition of your pipeline is a new deployment, and incurs costs.

Next Steps

After preparing your vectors, you can use the Retrieval API for chunks relevant to a query, or use the grounding module as part of an orchestration workflow for information retrieval and LLM interaction. For more information, see Retrieval API [page 504] or Using the Grounding Module [page 510].

You can add metadata to your pipeline, or update existing metadata by sending a patch request. For more information, see Update Metadata [page 464].

1.4.3.2.1.1.2.2.1.5 Create a Pipeline with SAP Build Work Zone

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- You have created a generic secret. For more information, see Generic Secrets for Grounding [page 398].
- You've added your grounding documents to your document store and you're using a supported data source and document type. For more information, see Pipelines API [page 438].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines`.

The following shows an example of a SAP Build Work Zone site to be indexed.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name> (destination)	The name of the generic secret created for SAP Build Work Zone

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
```

```
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "WorkZone",
  "metadata": {
    "destination": "<generic secret name>"
  }
}'
```

Scheduling updates

Optionally, you can add a `configuration` containing a `cronExpression` to schedule updates to your pipeline, at intervals of your choice.

For more information, see Cron Expressions [page 463].

For example:

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
    "type": "WorkZone",
    "metadata": {
        "destination": "<generic secret name>"
    },
    "configuration": {
        "cronExpression": "<cron expression>"
    }
}'
```

> **Note**
>
> Each repetition of your pipeline is a new deployment, and incurs costs.

Next Steps

After preparing your vectors, you can use the Retrieval API for chunks relevant to a query, or use the grounding module as part of an orchestration workflow for information retrieval and LLM interaction. For more information, see Retrieval API [page 504] or Using the Grounding Module [page 510].

Related Information

Integration With Document Grounding SAP Build Work Zone, Advanced Edition

1.4.3.2.1.1.2.2.1.6 Create a Pipeline with SAP Document Management Service

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- You have created a generic secret. For more information, see Generic Secrets for Grounding [page 398].
- You've added your grounding documents to your document store and you're using a supported data source and document type. For more information, see Pipelines API [page 438].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines`.

The following shows an example of a SAP Document Management service site to be indexed.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name> (destination)	The name of the generic secret created for SAP Document Management service

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "SDM",
  "metadata": {
    "destination": "<generic secret name>"
  }
}'
```

Scheduling updates

Optionally, you can add a `configuration` containing a `cronExpression` to schedule updates to your pipeline, at intervals of your choice.

For more information, see Cron Expressions [page 463].

For example:

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
    "type": "SDM",
    "metadata": {
        "destination": "sdm-secret-aicore"
    },
    "configuration": {
        "cronExpression": "<cron expression>"
    }
}'
```

> **Note**
>
> Each repetition of your pipeline is a new deployment, and incurs costs.

Next Steps

After preparing your vectors, you can use the Retrieval API for chunks relevant to a query, or use the grounding module as part of an orchestration workflow for information retrieval and LLM interaction. For more information, see Retrieval API [page 504] or Using the Grounding Module [page 510].

1.4.3.2.1.1.2.2.1.7 Create a Pipeline with ServiceNow

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- You have created a generic secret. For more information, see Generic Secrets for Grounding [page 398].
- You've added your grounding documents to your document store and you're using a supported data source and document type. For more information, see Pipelines API [page 438].
- If you're using metadata, you've configured a metadata server. For more information, see Prepare your Metadata API Server [page 416].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines`.

Without Metadata

The following shows an example of a ServiceNow site to be indexed.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name>	The name of the generic secret created for ServiceNow

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "ServiceNow",
  "configuration": {
    "destination": "<GENERIC_SECRET_NAME>"
  }
}'
```

With Metadata

Metadata is additional information associated with data that can help in identifying, categorizing, and retrieving relevant content.

The metadata attribute is optional. It accepts the destination name, which is used to connect to the ServiceNow metadata server to retrieve metadata for document indexing.

The metadata attribute should be used only if a metadata server is configured. To create a grounding pipeline without metadata, see Create a Document Grounding Pipeline Using the Pipelines API [page 440].

The following shows an example of a ServiceNow site to be indexed with metadata.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name> (destination)	The name of the generic secret created for ServiceNow
<generic secret name> (metadata)	The name of the generic secret created for the ServiceNow Metadata Server

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "ServiceNow",
  "configuration": {
    "destination": "<GENERIC_SECRET_NAME>"
  },
  "metadata": {
    "destination": "<METADATA_SECRET_NAME>"
  }
}'
```

With Scheduled Updates

`cronExpression` is optional. You can add it to your `configuration` to schedule updates to your pipeline, at intervals of your choice.

The following shows an example of a ServiceNow site to be indexed at regular intervals.

Populate the sample code with the following values:

Field	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<generic secret name>	The name of the generic secret created for ServiceNow
cronExpression	The cronExpression for the update frequency of your choice (optional). For more information, see Cron Expressions [page 463].

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "type": "ServiceNow",
  "configuration": {
    "destination": "<GENERIC_SECRET_NAME>",
    "cronExpression": "<CRON_EXPRESSION>"
  }
}'
```

Note

Each repetition of your pipeline is a new deployment, and incurs costs.

Next Steps

After preparing your vectors, you can use the Retrieval API for chunks relevant to a query, or use the grounding module as part of an orchestration workflow for information retrieval and LLM interaction. For more information, see Retrieval API [page 504] or Using the Grounding Module [page 510].

1.4.3.2.1.1.2.2.1.8 Create a Pipeline using a Metadata Configuration

Prerequisites

- You have created a resource group. For more information, see Create a Resource Group for Grounding [page 397].
- You have created a generic secret. For more information, see Generic Secrets for Grounding [page 398].
- You've created a metadata configuration and have noted the metadata configuration ID. For more information, see Create a Metadata Configuration [page 428] and List Metadata Configurations [page 430].

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines`.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Include the ID of your metadata configuration in your request. For example:

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines \
  --header 'AI-Resource-Group: <resource group>' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer $TOKEN'
  --data '{
    "type": "metadata",
    "configuration": {
      "metadataConfigId": "<metadata config id>"
     }
}'
```

1.4.3.2.1.1.2.2.1.9 Cron Expressions

You can schedule updates to your pipeline, at intervals of your choice by adding a `cronExpression` when you create your pipeline.

Cron expressions have five fields, represented by a string of asterisks. The values for these fields are as follows:

Pipeline Scheduling: Fields and Values

*	*	*	*	*
Minute (0–59)	Hour (0–23)	Day of the month (1–31)	Month (1–12)	Day of the week (0–6)

The minimum interval supported in the pipeline API is 1 hour.

Pipeline Scheduling Fields: Example Values

Expression	Schedule
0****	Every hour
00***	Every day at 12:00 a.m.
00**FRI	At 12:00 a.m. only on Friday
001**	At 12:00 a.m. on the first day of the month

> **Note**
>
> Each repetition of your pipeline is a new deployment, and incurs costs.

1.4.3.2.1.1.2.2.1.10 Update Metadata

Prerequisites

- You have created a pipeline using a repository that supports metadata. For more information, see Create a Document Grounding Pipeline Using the Pipelines API [page 440].

Context

Metadata is additional information associated with data that can help in identifying, categorizing, and retrieving relevant content. You can use your metadata values later to search your pipeline. For more information, see Search a Pipeline [page 476].

Procedure

For repository types that support metadata, you can add or update your metadata by sending a PATCH request to to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}`.

Include the pipeline ID for the pipeline that you want to update.

Ensure that you've the following headers set:

Header	Value
Authorization	Bearer $AUTH_TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. This URL can also be set as an environment variable.

Metadata organizes information using categories (keys) paired with their related values.

> **Sample Code**

```
curl --request PATCH \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}} \
  --header 'ai-resource-group: <resource_group>' \
  --header 'authorization: Bearer $AUTH_TOKEN' \
```

```
  --header 'content-type: application/json' \
  --data '{
  "metadata": {
    "dataRepositoryMetadata": [
      {
        "key": "pipeline/purpose",
        "value": [
          "patch-pipeline1"
        ]
      },
      {
        "key": "pipeline/owner",
        "value": [
          "data-engineering",
          "analytics-team"
        ]
      }
    ]
  }
}'
```

1.4.3.2.1.1.2.2.2 Data Pipelines

You can get the details of your data pipelines using the following endpoints:

Get All Pipelines

Use this endpoint to retrieve all document grounding pipelines created in your specified AI resource group.

Each pipeline represents a configured end-to-end process that:

- Fetches documents from a supported data source
- Preprocesses and chunks the content
- Stores semantic embeddings into the HANA Vector Store

This is useful when you want to:

- Review existing pipelines before creating or executing a new one
- Manage pipeline inventory programmatically
- Support dashboards, monitoring, or integration flows
- Enable pipeline reusability across projects

> **Note**
>
> If you use the Pipeline APIs for creating and retrieving Pipeline details, you do not need to use the Vector API separately, it's already handled as part of the pipeline execution. After a pipeline runs, you can use the Retrieval API to query grounded content.
>
> The endpoint for retrieving document pipelines is `$AI_API_URL/v2/lm/document-grounding/pipelines`

You can use the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines` to retrieve all document grounding pipelines.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines\
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

This endpoint returns a response containing an array of pipeline objects, in JSON format.

Available statuses for pipelines are as follows:

- NEW
- UNKNOWN
- INPROGRESS
- FINISHED
- FINISHEDWITHERRORS
- TIMEOUT

Pagination Support

You can use OData-compliant query parameters to manage large result sets:

Parameter	Type	Description
$top	Integer	Limits the number of results returned. for example., $top=10 returns 10 results.
$skip	Integer	Skips a specific number of results. for example, $skip=20 skips the first 20.
$count	Boolean	If true, includes the total count in the response. If false, includes the total number of records in the response.
metaConfigId	String	Filter pipelines by metadata configuration ID.

For example:

Sample Code

```
curl --request GET
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines?
$top=10&$skip=20&$count=true"
  --header "AI-Resource-Group: {{resource_group}}"
  --header 'Authorization: Bearer {{access_token}}'
```

or:

```
curl --request GET
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines?
metadataConfigId=<yourMetadataConfigId>"
  --header "AI-Resource-Group: {{resource_group}}"
  --header 'Authorization: Bearer {{access_token}}'
```

Successful responses include response code **200** and a filtered list of pipelines.

Get Pipeline Details

This endpoint lets you fetch full details of a specific document grounding pipeline by using its unique pipelineId.

It is useful when you want to:

- Inspect the pipeline's configuration (fore example, data source, chunking strategy, schedule)
- Debug or validate how a pipeline is set up
- Display pipeline metadata
- Get last updated time and status before reusing or modifying a pipeline

You can get pipeline details using the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}`.

Populate the code snippet with the following:

- `{{resource_group}}`: the ID of the specific pipeline you want to retrieve.
- `AI_API_URL`: the base URL of your SAP AI Core environment.
- `{{access_token}}`: Your access token for SAP AI Core
- `{{pipelineId}}`: the AI resource group ID assigned to your account.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}} \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

This endpoint returns a response containing details of the pipeline configuration and metadata, in JSON format.

Available statuses for pipelines are as follows:

- NEW

- UNKNOWN
- INPROGRESS
- FINISHED
- FINISHEDWITHERRORS
- TIMEOUT

Get Pipeline Status

This endpoint allows you to retrieve the current status of a specific document grounding pipeline. It provides information on the lifecycle state, readiness, or any ongoing processing of the pipeline.

Populate the code snippet with the following:

- `{{resource_group}}`: the ID of the specific pipeline you want to retrieve..
- `AI_API_URL`: the base URL of your SAP AI Core environment.
- `{{access_token}}`: Your access token for SAP AI Core
- `{{pipelineId}}`: the AI resource group ID assigned to your account.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/status \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

This endpoint returns a response containing details of the pipeline status, in JSON format.

Available statuses for pipelines are as follows:

- NEW
- UNKNOWN
- INPROGRESS
- FINISHED
- FINISHEDWITHERRORS
- TIMEOUT

Delete a Pipeline

Use this endpoint to permanently delete a specific document grounding pipeline by its unique pipelineId. This is typically done when a pipeline is no longer required, misconfigured, or needs to be recreated with different settings. After deletion, the pipeline and its configuration are irreversibly removed.

You can delete a pipeline using the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}`.

Populate the code snippet with the following:

- `{{resource_group}}`: the ID of the specific pipeline you want to retrieve..
- `AI_API_URL`: the base URL of your SAP AI Core environment.
- `{{access_token}}`: Your access token for SAP AI Core
- `{{pipelineId}}`: the AI resource group ID assigned to your account.

Sample Code

```
curl --request DELETE \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}} \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

This endpoint returns a response containing confirmation of deletion, in JSON format.

1.4.3.2.1.1.2.2.3 Executions

You can get the details of your executions in data pipelines using the following endpoints:

List All Executions of a Document Grounding Pipeline

This endpoint allows you to list all executions associated with a specific document grounding pipeline. The response includes execution metadata such as status, timestamps, and identifiers.

You can list all the executions by using the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/executions?lastExecution=false`.

Populate the code snippet with the following:

- `{{pipelineId}}`: The ID of the pipeline whose executions you want to list
- `lastExecution`: Set to `true` to fetch only the latest execution (the default value is `false`)
- `{{resource_group}}`: The AI resource group ID assigned to your account
- `{{access_token}}`: Your access token for SAP AI Core

This endpoint returns a response in JSON format.

Sample Code

```
curl --request GET \
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/
executions?lastExecution=false" \
  --header "AI-Resource-Group: {{resource_group}}"
  --header 'Authorization: Bearer {{access_token}}'
```

Pagination Support

You can use OData-compliant query parameters to manage large result sets:

Parameter	Type	Description
`$top`	Integer	Limits the number of results returned. For example, `$top=10` returns 10 results.
`$skip`	Integer	Skips a specific number of results. For example, `$skip=20` skips the first 20.
`$count`	Boolean	If `true`, includes the total count in the response. If `false`, includes the total number of records in the response.

Example

This code returns a JSON array of pipeline execution objects.

```
curl --request GET
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/
executions?$top=10&$skip=20&$count=true"
  --header "AI-Resource-Group: {{resource_group}}"
  --header 'Authorization: Bearer {{access_token}}'
```

Available statuses for executions are as follows:

- NEW
- UNKNOWN
- INPROGRESS
- FINISHED
- FINISHEDWITHERRORS
- TIMEOUT

Get Specific Execution Details

This endpoint allows you to retrieve the details of a specific execution of a document grounding pipeline by providing the pipeline ID and execution ID.

You can retrieve a specific execution by using the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/executions/{{executionId}}`.

Populate the code snippet with the following:

- `{{pipelineId}}`: The ID of the pipeline that contains the execution that you want to retrieve
- `{{executionId}}`: The ID of the specific execution
- `{{resource_group}}`: The AI resource group ID assigned to your account
- `{{access_token}}`: Your access token for SAP AI Core

This endpoint returns a response in JSON format. The response includes fields such as `status`, `startTime`, `endTime`, and any `errorMessage`.

Sample Code

```
curl --request GET \
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/
executions/{{executionId}}" \
  --header "AI-Resource-Group: {{resource_group}}"
  --header 'Authorization: Bearer {{access_token}}'
```

Available statuses for executions are as follows:

- NEW
- UNKNOWN
- INPROGRESS
- FINISHED
- FINISHEDWITHERRORS
- TIMEOUT

1.4.3.2.1.1.2.2.4 Documents

You can get the details of your documents in data pipelines using the following endpoints:

List All Documents for an Execution

This endpoint allows you to retrieve all documents associated with a specific execution, including the processing statuses of the documents. You must provide the pipeline ID and the execution ID as parameters.

You can list all documents for an execution by using the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/executions/{{executionId}}/documents`.

Populate the code snippet with the following:

- `{{pipelineId}}`: The pipeline ID
- `{{executionId}}`: The ID of the specific execution
- `{{resource_group}}`: The AI resource group ID assigned to your account

This endpoint returns a response in JSON format. The response includes the list of all documents processed during the specified execution.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/
executions/{{executionId}}/documents \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

Available statuses for documents are as follows:

- TO_BE_PROCESSED
- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED_TO_BE_RETRIED
- TO_BE_SCHEDULED

Pagination Support

You can use OData-compliant query parameters to manage large result sets:

Parameter	Type	Description
`$top`	Integer	Limits the number of results returned. For example, `$top=10` returns 10 results.
`$skip`	Integer	Skips a specific number of results. For example, `$skip=20` skips the first 20.
`$count`	Boolean	If `true`, includes the total count in the response. If `false`, includes the total number of records in the response.

Example

This code returns a JSON array of document objects associated with the pipeline execution.

```
curl --request GET
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/
executions{{executionId}}/documents?$top=10&$skip=20&$count=true"
  --header "AI-Resource-Group: {{resource_group}}" \
  --header 'Authorization: Bearer {{access_token}}'
```

Available statuses for documents are as follows:

- TO_BE_PROCESSED
- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED_TO_BE_RETRIED
- TO_BE_SCHEDULED

Get a Specific Document from an Execution

This endpoint allows you to retrieve detailed information for a specific document within an execution, including its processing status. The endpoint requires the input parameters for the pipeline ID, execution ID, and document ID.

You can retrieve a specific document by using the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/executions/{{executionId}}/documents/{{documentID}}`.

Populate the code snippet with the following:

- `{{pipelineId}}`: The pipeline ID
- `{{executionId}}`: The ID of the specific execution
- `{{documentId}}`: The ID of the specific document
- `{{resource_group}}`: The AI resource group ID assigned to your account
- `{{access_token}}`: Your access token for SAP AI Core

This endpoint returns a response in JSON format. The response includes the specific document processed during the specified execution. It also includes the document's processing status by pipeline ID, execution ID, and document ID.

Sample Code

```
curl --request GET \
  --url $AI_API_URL /v2/lm/document-grounding/pipelines/{{pipelineId}}/
executions/{{executionId}}/documents/{{documentID}} \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

Available statuses for documents are as follows:

- TO_BE_PROCESSED
- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED_TO_BE_RETRIED
- TO_BE_SCHEDULED

List All Documents for a Pipeline

This endpoint allows you to retrieve all documents for a pipeline, including the processing statuses by pipeline ID.

You can retrieve all documents by using the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/documents`.

Populate the code snippet with the following:

- `{{pipelineId}}`: The pipeline ID

- `{{resource_group}}`: The AI resource group ID assigned to your account
- `{{access_token}}`: Your access token for SAP AI Core

This endpoint returns a response in JSON format.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/
documents \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

Available statuses for documents are as follows:

- TO_BE_PROCESSED
- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED_TO_BE_RETRIED
- TO_BE_SCHEDULED

Pagination Support

You can use OData-compliant query parameters to manage large result sets:

Parameter	Type	Description
`$top`	Integer	Limits the number of results returned. For example, `$top=10` returns 10 results.
`$skip`	Integer	Skips a specific number of results. For example, `$skip=20` skips the first 20.
`$count`	Boolean	If `true`, includes the total count in the response. If `false`, includes the total number of records in the response.

Example

This code returns a JSON array of document objects associated with the pipeline execution.

```
curl --request GET
  --url "$AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/
documents?$top=10&$skip=20&$count=true"
  --header "AI-Resource-Group: {{resource_group}}"
  --header 'Authorization: Bearer {{access_token}}'
```

Get a Specific Document for a Pipeline

This endpoint allows you to retrieve detailed information about a document within a specific pipeline, including the document's processing status. The endpoint requires the pipeline ID and document ID as parameters.

Populate the code snippet with the following:

- `{{pipelineId}}`: The pipeline ID
- `{{documentId}}`: The ID of the specific document
- `{{resource_group}}`: The AI resource group ID assigned to your account
- `{{access_token}}`: Your access token for SAP AI Core

This endpoint returns a response in JSON format. The response includes the specific document processed in the pipeline.

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/{{pipelineId}}/
documents/{{documentID}}
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

Available statuses for documents are as follows:

- TO_BE_PROCESSED
- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED_TO_BE_RETRIED
- TO_BE_SCHEDULED

1.4.3.2.1.1.2.2.5 Manually Restart a Document Grounding Pipeline

This endpoint allows you to retrain an existing pipeline manually.

Context

You can start an existing pipeline by sending a POST request to the endpoint: `/v2/lm/document-grounding/pipelines/trigger` and including your pipeline ID in your request.

Procedure

1. Populate the sample code with the following values:

	Value
`{{resource_group}}`	The AI resource group assigned to your account
`$AI_API_URL`	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
`{{access_token}}`	Your access token for SAP AI Core
`{{pipelineId}}`	The ID of the pipeline that you want to start
`metadataOnly`	A Boolean value. `true` updates the metadata only. `false` updates you semantic embeddings by processing the documents in your repository.

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/pipelines/trigger \
  --header 'ai-resource-group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
  --header 'content-type: application/json' \
  --data '{
    "pipelineId":"{{pipelineId}}",
    "metadataOnly": "<Boolean>"
  }'
```

Results

The response includes a response code. Code 202 is a successful response.

1.4.3.2.1.1.2.2.6 Search a Pipeline

Prerequisites

You've created a document grounding pipeline, and included your choice of metadata in the `dataRepositoryMetadata` field. For more information, see Create a Document Grounding Pipeline Using the Pipelines API [page 440].

Procedure

Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines/search`. Include your data repository key:value pair in your request.

Set the following in the headers:

- `AI_API_URL`: the base URL of your SAP AI Core environment.
- `{{resource_group}}`: The AI resource group ID or document grounding management
- `{{access_token}}`: Your access token for SAP AI Core

For example:

```
curl --request POST \
--url $AI_API_URL/v2/lm/document-grounding/pipelines/search \
--header 'AI-Resource-Group: {{resource_group}}'
--data '{
    "dataRepositoryMetadata": [
        {
            "key": "example",
            "value": [
                "demo"
            ]
        }
    ]
}'
```

The output lists any pipelines that match the data repository metadata.

Output Code

```
{
    "count": 1,
    "resources": [
        {
            "id": "<pipelineid>",
            "status": "<status-of-pipeline-execution>",
            "type": "<type-of-repository>",
            "configuration": {
                "sharePoint": {
                    "id": "<site-id>",
                    "name": "<site-name>"
                }
            }
        }
    ]
}
```

1.4.3.2.1.1.2.3 Vector API

Vector API is a set of REST APIs used for document ingestion and retrieval using vector embeddings.

You can use the Vector API to:

- Create collections.
- Create documents by directly using chunks of data.

- Store data in a vector database.
- Provide repository IDs to access the data.

The Vector API incorporates the following units:

Collections: A logical container used to store and manage embedded documents.

Documents: A unit of text-based content that you upload into a collection, to be used for semantic search and grounding.

Chunks: A small segment of a document's text, used in the Vector API to improve the accuracy and efficiency of semantic search.

Capacity Quotas and Input Limits

Vector Ingestion – Capacity Limits

The following capacity limits apply:

Category	Description	Limit Type	Limit
Collections	Max collections	Quota	55,000
Documents	Max documents per collection	Quota	100,000
Documents	Max documents per ingestion request	Input Limit	100
Chunks	Max chunks per document	Quota	200,000

Metadata Limits (Ingestion and Query Time)

The following metadata limits apply:

Metadata Type	Limit Type	Limit
Collection Metadata	Quota / Input Limit	500 / 5,000
Document Metadata	Quota / Input Limit	500 / 5,000
Chunk Metadata	Quota / Input Limit	500 / 5,000

Vector Search – Query Limits

The following query limits apply:

Category	Description	Limit Type	Limit
Filters	Max filters per query	Input Limit	15
Collections	Max collection IDs per query	Input Limit	5,000
Results	Max chunk results per query	Input Limit	100
Results	Max document results per query	Input Limit	10

1.4.3.2.1.1.2.3.1 Preparing Data Using the Vector API

Vector API is a microservice provided with a Rest API and endpoints for creating and managing collection and documents.

Prerequisites

You have created a resource group for grounding purposes. For more information, see Create a Resource Group for Grounding [page 397].

To insert your document chunks into the vector store, you will need to follow the following operations:

1. Create a collection For more information see Create a Collection [page 480].
2. Insert documents and chunks into the collection. For more information see Create a Document [page 487].

API Request Format

When you build your API requests, follow the format outlined in the *Vector* section of the Grounding API. For more information, see Grounding API.

> ⚠ Caution
>
> Do not expose personally identifiable or privileged information in your documents or chunks when using generative AI hub. Personally identifiable information is any data that can be used alone, or in combination, to identify the person that the data refers to. The document grounding capability has no mechanism for determining the type of data that it processes, and does not filter confidential or other privileged information.

Next Steps

After preparing your vectors, you can use the Retrieval API for chunks relevant to a query, or use the grounding module as part of an orchestration workflow for information retrieval and LLM interaction. For more information, see Retrieval API [page 504] or Using the Grounding Module [page 510].

Alternatively, you can search specific collections using vector search. For more information, see Vector Search [page 498]

1.4.3.2.1.1.2.3.1.1 Collections

1.4.3.2.1.1.2.3.1.1.1 Create a Collection

This endpoint allows you to create a new collection. Collections are logical containers used to store and manage embedded documents and their associated chunks.

Prerequisites

You have created a resource group for grounding purposes. For more information, see Create a Resource Group for Grounding [page 397]

Context

This endpoint allows you to create a new collection for storing documents and their associated chunks.

Procedure

1. Send a POST request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections`.
   Populate the sample code with the following values:

Placeholder	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<title of the collection>	Your choice of title for the collection
<your choice of valid uuid>	Your choice of UUID that can be used later to locate the collection
<embedding-model-name>	The name of the embedding model to be used during vectorization
metadata.purpose	Your choice of text about the purpose of the collection
metadata <key> <value> pair	Your choice of metadata key value pair

> **Note**
>
> The `id`, `title` and `metadata` fields are optional.

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "id": "<your choice of valid uuid>",
  "title": "<title of the collection>",
  "embeddingConfig": {
    "modelName": "<embedding-model-name>"
  },
  "metadata": [
    {
      "key": "purpose",
      "value": [
        "<text>"
      ]
    },
    {
      "key": "<a-random-key>",
      "value": [
        "<text>"
      ]
    }
  ]
}'
```

Results

The response is returned in JSON format, and includes a `location` header that contains the endpoint that can be used to get the status of collection creation.

Next Steps

Check the status of your collection. For more information, see Get Collection Creation Status [page 484].

1.4.3.2.1.1.2.3.1.1.2 Get Collection Details

This endpoint allows you to retrieve the details of a specific collection using its collection ID.

Procedure

Send a GET request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}`.

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
{{collectionId}}	The ID of the collection

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections/
{{collectionId}} \
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes details of your collection.

1.4.3.2.1.1.2.3.1.1.3 Get Collections

This endpoint allows you to retrieve a list of all collections associated with a specific resource group.

Procedure

Send a GET request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections`

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections \
  --header 'AI-Resource-Group: {{resource_group}}'
```

```
  --header 'Authorization: Bearer {{access_token}}'
```

Pagination Support

You can use OData-compliant query parameters to manage large result sets:

Parameter	Type	Description
$top	Integer	Limits the number of results returned. for example., $top=10 returns 10 results.
$skip	Integer	Skips a specific number of results. for example, $skip=20 skips the first 20.
$count	Boolean	If true, includes the total count in the response. If false, includes the total number of records in the response.

For example:

```
curl --request GET
--url "$AI_API_URL/v2/lm/document-grounding/vector/collections?
$top=10&$skip=20&$count=true"
--header "AI-Resource-Group: {{resource_group}}"
--header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes a JSON array of collection objects.

1.4.3.2.1.1.2.3.1.1.4 Get a Collection

This endpoint allows you to retrieve details of a specific collection within a specified resource group.

Procedure

Send a GET request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections`
Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account

	Value
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
{{collectionId}}	The ID of the collection

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections/
{{collectionId}} \
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes details of your collection.

1.4.3.2.1.1.2.3.1.1.5 Get Collection Creation Status

This endpoint allows you to retrieve the status of a collection creation process using the collection ID.

Procedure

Send a GET request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections/{{id}}/creationStatus`.

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
{{id}}	The ID of the collection

Sample Code

```
curl --request GET \
```

```
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections/{{id}}/
creationStatus \
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes a the status of your collection creation.

Next Steps

After the collection has been created succesfully, you can add documents to your collection. For more information, see Create a Document [page 487].

1.4.3.2.1.1.2.3.1.1.6 Delete a Collection

This endpoint is used to delete a collection using its collection id.

Procedure

Send a DELETE request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}`

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
{{collectionId}}	The ID of the collection

Sample Code

```
curl --request DELETE \
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections/
{{collectionId}} \
```

```
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes a `location` header that contains the endpoint that can be used to get the status of deletion. Successful requests return a 202 response.

1.4.3.2.1.1.2.3.1.1.7 Get Collection Deletion Status

This endpoint allows you to retrieve the status of a collection deletion using the collection ID.

Procedure

Send a GET request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections/{{id}}/deletionStatus`

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
{{id}}	The ID of the collection

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections/{{id}}/
deletionStatus \
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes a the status of your collection deletion.

1.4.3.2.1.1.2.3.1.2 Documents

1.4.3.2.1.1.2.3.1.2.1 Create a Document

This endpoint allows you to add a document with associated metadata and chunks to a collection. Documents are units of text-based content that you upload into a collection, to be used for semantic search and grounding.

Prerequisites

You have created a document collection. For more information, see Vector API [page 477] and Create a Collection [page 480].

You know the ID of the collection that you want to add documents to. For more information, see Get a Collection [page 483].

Procedure

1. Send a POST request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents`.
   Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
Content-Type	application/json
{{collectionId}}	The ID of the collection
metadata <key> <value> pair	Your choice of metadata key value pair
chunks	Text content in chunks
metadata <key> <value> pair (chunks)	Your choice of metadata key value pair

Sample Code

```
curl --request POST \
 --url  $AI_API_URL/v2/lm/document-grounding/vector/collections/
{{collectionId}}/documents\
 --header 'AI-Resource-Group: {{resource_group}}' \
 --header 'Content-Type: application/json' \
 --header 'Authorization: Bearer {{access_token}}'
  --data '{
```

```
  "documents": [
    {
      "metadata": [
        {
          "key": "url",
          "value": [
            "http://hello.com",
            "123"
          ]
        }
      ],
      "chunks": [
        {
          "content": "<chunk content 1>",
          "metadata": [  //
            {
              "key": "index",
              "value": [
                "1"
              ]
            }
          ]
        },
        {
          "content": "<chunk content 2>",
          "metadata": [
            {
              "key": "index",
              "value": [
                "2"
              ]
            }
          ]
        }
      ]
    }
  ]
}'
```

Results

The response is returned in JSON format, and includes a response code.

The response codes are:

- 201: Created (successful response)
- 400: The specification of the resource was incorrect
- 404: The specification of the resource was incorrect
- 422: There are validation issues with the data

Next Steps

To search for relevant chunks within a specific collection or across all collections based on a user query, use vector search. For more information, see Vector Search [page 498].

To search for relevant chunks across all collections based on a user query, use the Retrieval API. For more information, see Retrieval API [page 504].

To use your documents as part of an orchestration workflow, see Using the Grounding Module [page 510].

1.4.3.2.1.1.2.3.1.2.2 Update a Document

This endpoint allows you to update an existing document within a collection, including its associated metadata and chunks.

Procedure

Send a PATCH request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents`

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
Content-Type	application/json
{{collectionId}}	The ID of the collection
{{documentId}}	The ID of the document
metadata <key> <value> pair	Your choice of metadata key value pair
chunks	Text content in chunks
metadata <key> <value> pair (chunks)	Your choice of metadata key value pair

Sample Code

```
curl --request PATCH \
 --url $AI_API_URL/v2/lm/document-grounding/vector/collections/
{{collectionId}}/documents \
 --header 'AI-Resource-Group: {{resource_group}}' \
 --header 'Content-Type: application/json' \
 --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "documents": [
    {
      "id": {{documentId}},
      "metadata": [
        {
```

```
          "key": "url",
          "value": [
            "http://hello.com",
            "123"
          ]
        }
      ],
      "chunks": [
        {
          "content": "<This is an updated document chunk content>",
          "metadata": [
            {
              "key": "index",
              "value": [
                "1"
              ]
            }
          ]
        }
      ]
    }
  ]
}'
```

Results

The response is returned in JSON format, and includes a response code.

The reponse codes are:

- 200: OK (successful response)
- 400: The specification of the resource was incorrect
- 404: The specification of the resource was incorrect
- 422: There are validation issues with the data

1.4.3.2.1.1.2.3.1.2.3 Get All Documents in a Collection

This endpoint allows you to retrieve all documents associated with a specified collection ID.

Procedure

Send a GET request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents`

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
{{collectionId}}	The ID of the collection

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections/
{{collectionId}}/documents \
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
```

Pagination Support

You can use OData-compliant query parameters to manage large result sets:

Parameter	Type	Description
$top	Integer	Limits the number of results returned. for example., $top=10 returns 10 results.
$skip	Integer	Skips a specific number of results. for example, $skip=20 skips the first 20.
$count	Boolean	If true, includes the total count in the response. If false, includes the total number of records in the response.

For example:

```
curl --request GET
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/
documents?$top=10&$skip=20&$count=true"
  --header "AI-Resource-Group: {{resource_group}}"
  --header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes a JSON array of collection document objects.

1.4.3.2.1.1.2.3.1.2.4 Get A Document

This endpoint allows you to retrieve a specific document from a collection using the collection ID and document ID.

Procedure

Send a GET request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents/{{documentId}}`

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
{{collectionId}}	The ID of the collection
{{documentId}}	The ID of the document

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections/
{{collectionId}}/documents/{{documentId}} \
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes details of your document.

1.4.3.2.1.1.2.3.1.2.5 Delete a Document

This endpoint allows you to delete a specific document from a collection using the collection ID and document ID.

Procedure

Send a DELETE request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/collections/{{collectionId}}/documents/{{documentId}}`

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
{{collectionId}}	The ID of the collection
{{documentId}}	The ID of the document

Sample Code

```
curl --request DELETE \
  --url $AI_API_URL/v2/lm/document-grounding/vector/collections/
{{collectionId}}/documents/{{documentId}} \
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes a response code.

The response codes are:

- 202: Deleted (successful response)
- 400: The specification of the resource was incorrect
- 404: The specification of the resource was incorrect
- 422: There are validation issues with the data

1.4.3.2.1.1.2.3.1.2.6 Delete Documents

This endpoint allows you to bulk delete specific documents using document IDs.

Procedure

Send a DELETE request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/documents`
Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<id1>..	The Ids of the documents that you want to delete.

Include the IDs of the documents that you want to delete in the `ids` field of the data.

Sample Code

```
curl --request DELETE \
  --url $AI_API_URL/v2/lm/document-grounding/vector/documents \
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
    "ids": [
      "<id1>",
      "<id2>",
      "<id3>"
    ]
}'
```

Results

The response is returned in JSON format, and includes a response code.

The response codes are:

- 202:
  - Deleted - document IDs successfully deleted
  - NotFound – document IDs not found and not deleted
- 400: The specification of the resource was incorrect
- 404: The specification of the resource was incorrect
- 422: There are validation issues with the data

1.4.3.2.1.1.2.3.2 Metadata

Context

Metadata is additional information, such as tags, categories, or custom attributes that you can add to your collections, documents, or chunks to help organize, filter, and search your content more effectively.

You can update metadata for multiple collections or multiple documents or multiple chunks in a single API request.

You can use the following operations:

Operation	Description
`add`	If the key is present, then it will add the new values. If the key is absent, it will add the key and the values.
`remove`	If the key is present, then it will remove the values from it.
`replace`	If the key is present, it will replace the entire list of values with the new values. If the key is absent, it will add the key and the values.
`delete_key`	If the key is present, then it will remove the key and all its values.

Limits

The following limits are the maximum number of updates that can be applied in each update:

- **5** update blocks per **PATCH request**. Ids can be grouped as required.
- **10** IDs per **update block**.
- **10** metadata updates per **update block**.
- For `remove` or `delete_key` operations: **1** operation per **update**.
- For `add` or `replace` operations: **2** operations per **update**.
- **10** values per **operation**.

Procedure

Send a PATCH request to the respective endpoint

Resource to be Updated	Endpoint
Collections	`$AI_API_URL/v2/lm/document-grounding/vector/collections/metadata`

Resource to be Updated	Endpoint
Documents	`$AI_API_URL/v2/lm/document-grounding/vector/documents/metadata`
Chunks	`$AI_API_URL/v2/lm/document-grounding/vector/chunks/metadata`

Populate the sample code with the following values:

Parameter	Description
`{{resource_group}}`	The AI resource group assigned to your account
`$AI_API_URL`	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
`{{access_token}}`	Your access token for SAP AI Core
`{{resource_id_1}}`	The ID of the collection, document, or chunk that you want to update. You can update multiple resources by providing a list of IDs.
`{{resource_id_2}}` (optional)	The ID of a second collection, document, or chunk that you'd like to update in the same request.

The sample provided completes the following tasks:

For `{{resource_id_1}}`:

- If the metadata key `"department"` is not present, the key will be added with values `"finance"` and `"legal"`.
- If the metadata key `"department"` is present, the values `"finance"` and `"legal"` will be added to the existing values.
- If the metadata key `"department"` is present, the value `"tech"` will be removed from the existing values.
- If the metadata key `"status"` exists, the key will be removed.

For `{{resource_id_2}}`:

- If the metadata key `"department"` is present, all of the values associated with the key will be replaced by the two values `"finance"` and `"legal"`.
- If the metadata key `"department"` is not present, the key will be added with values `"finance"` and `"legal"`

Sample Code

```
curl --request PATCH \
 --url $AI_API_URL/v2/lm/document-grounding/vector/collections/metadata \
 --header 'AI-Resource-Group: {{resource_group}}' \
 --header 'Content-Type: application/json' \
 --header 'Authorization: Bearer {{access_token}}'
  --data '{
    "updates": [
        {
            "ids": [
                {{resource_id_1}}
            ],
            "metadataUpdates": [
                {
```

```
                    "key": "department",
                    "operations": [
                        {
                            "op": "add",
                            "values": [
                                "finance",
                                "legal"
                            ]
                        },
                        {
                            "op": "remove",
                            "values": [
                                "tech"
                            ]
                        }
                    ]
                },
                {
                    "key": "status",
                    "operations": [
                        {
                            "op": "delete_key"
                        }
                    ]
                }
            ]
        },
        {
            "ids": [
                {{recource_id_2}}
            ],
            "metadataUpdates": [
                {
                    "key": "department",
                    "operations": [
                        {
                            "op": "replace",
                            "values": [
                                "finance",
                                "legal"
                            ]
                        }
                    ]
                }
            ]
        }
    ]
}'
```

Results

The response is returned in JSON format, and includes details of the updated resources and metadata.

1.4.3.2.1.1.2.3.3 Vector Search

This endpoint allows you to perform a search for relevant chunks within a specific collection or across all collections based on a user query. The response includes chunks that match the query, filtered by collection and document metadata.

Metadata Filtering Overview

Metadata Filtering lets you retrieve and filter documents, collections, and chunks with advanced metadata queries. It supports flexible, multi-level filtering to give you control over search results.

> ⓘ Note
>
> Filtering is available at the collection, document, and chunk levels.

The following are the match modes supports :

Key Match Modes

Simple Matching

- Provide `key-value` pairs in the `"chunkMetadata"`, `"documentMetadata"`, or `"collectionMetadata"` fields.
- Each entry defines a metadata key and a list of valid values.
- All specified `key-value` pairs must match for an item to be returned (logical **AND** across keys).
- Items containing all the given metadata values are included in the results.

Complex Matching

- Define `key-value` pairs using nested logical expressions in the filter field.
- Use logical operators (**AND**, **OR**) to combine multiple conditions.
- Each condition can define the metadata key, values, and the scope (collection, document, or chunk).
- Filters can be nested to create more advanced queries.

> → Tip
>
> Complex filtering is recommended because it offers greater flexibility and is the recommended approach.

Values Match Modes

Match Modes

- **ANY**
  - At least one metadata value matches.
  - This is the default match mode.
  - Defined only at injection time.
- **ALL**
  - All document metadata values must match exactly, or be a subset of the search metadata.
  - Supported only at the document level.

- Defined only at injection time.

Select Modes

- Supports `ignoreIfKeyAbsent` to control how missing keys are handled. This option can be set at query time and is supported only at the document level.
- The default is `False`.
- When set to `True`, documents that do not contain the specified key are also included in the results.

Procedure

Send a POST request to the endpoint: `$AI_API_URL/v2/lm/document-grounding/vector/search`

Populate the sample code with the following values:

Parameter	Description
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
query	Your user query
collectionIds	Enter `"*"` to search all collections or add a collectionId to search a specific collection. To search more than one specific collection, add `collectionIds` in a comma separated list.
documentMetadata	Filters based on metadata previously assigned

Example with Simple Matching

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/vector/search \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "query": "is Joule an AI Copilot?",
  "filters": [
    {
      "id": "String",
      "collectionIds": ["*"],
      "configuration": {},
      "collectionMetadata": [],
      "documentMetadata": [
        {
          "key": "url",
          "value": [
```

```
            "http://hello.com",
            "1234"
          ]
        }
      ],
      "chunkMetadata": []
    }
  ]
}'
```

Example with Complex Matching

Use the following examples to define grounding metadata filters. The example below shows a filter that retrieves content matching the following metadata specification:

- `key1=["value1"]` OR `key2=["value2"]`
- `(key1=["value1"]` AND `key2=["value2"])` OR `key3=["value3"]`
- `(key1=["value1"]` AND `key2=["value2"])` OR `key3=["value3"]`
- `key1=["value1"]` AND `key2=["value2"]`

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/vector/search \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "query": "is Joule an AI Copilot?",
  "filters": [
    {
      "id": "String",
      "collectionIds": ["*"],
      "configuration": {},
      "collectionMetadata": [],
      "documentMetadata": [],
      "chunkMetadata": [],
      "filter": {
        "operator": "and",
        "left": {
          "key": "key1",
          "value": [
          "value1"
          ],
          "scope": "document"
        },
        "right": {
          "key": "key2",
          "value": [
          "value2"
          ],
          "scope": "document"
        }
      }
    }
  ]
}'
```

Nested Complex Matching

The following example shows a complex filter with nested conditions. It retrieves all content where the chunk has `key1=value1`, and either the document has `key2=value2` or the collection has `key3=value3`.

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/vector/search \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "query": "is Joule an AI Copilot?",
  "filters": [
    {
      "id": "String",
      "collectionIds": ["*"],
      "configuration": {},
      "collectionMetadata": [],
      "documentMetadata": [],
      "chunkMetadata": [],
      "filter": {
        "operator": "and",
        "left": {
          "key": "key1",
          "value": [
            "value1"
          ],
          "scope": "chunk"
        },
        "right": {
          "operator": "or",
          "left": {
            "key": "key2",
            "value": [
              "value2"
            ],
            "scope": "document"
          },
          "right": {
            "key": "key3",
            "value": [
              "value3"
            ],
            "scope": "collection"
          }
        }
      }
    }
  ]
}'
```

Nested Complex Filter with Match Mode ALL

The document is created with metadata where `key1=["value11"]` uses the default match mode **ANY**, and `key2=["value21","value22"]` uses match mode **ALL**. The search request contains both keys at the document level.

Because the document's`key2` contains the full set of requested values, the document matches the filter and is returned.

Sample Code

```
curl --request POST \
  --url $AI_API_URL/v2/lm/document-grounding/vector/search \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer {{access_token}}'
  --data '{
  "query": "is Joule an AI Copilot?",
  "filters": [
    {
      "id": "String",
      "collectionIds": ["*"],
      "configuration": {},
      "collectionMetadata": [],
      "documentMetadata": [],
      "chunkMetadata": [],
      "filter": {
        "operator": "and",
        "left": {
          "key": "key1",
          "value": [
          "value11"
          ],
          "scope": "document"
        },
        "right": {
          "key": "key2",
          "value": [
          “value21”, "value22", “value23”
          ],
          "scope": "document"
        }
      }
    }
  ]
```

The response is returned in JSON format, and includes relevant chunks based on similarity search. Each chunk object has a `searchScores` object, describing how relevant the chunk is to the search query.

The results also contain a `denseRetrievalScore` containing the cosine similarity score of the text embeddings.

Payload Attributes

Attribute	Type	Required	Description	Constraints
query	string	Yes	The search query text	maxLength: 2000, minLength: 1
filters	array	Yes	Array of search filters	N/A
filters[].id	string	Yes	Unique identifier for this search filter	N/A

Attribute	Type	Required	Description	Constraints
filters[].collectionIds	array of strings	Yes	List of collection IDs to search in. Use ["*"] to search all collections	N/A
filters[].configuration	object	Yes	Search configuration options	N/A
filters[].configuration.maxChunkCount	integer	No	Maximum number of chunks to return	> 0, cannot be used with maxDocumentCount
filters[].configuration.maxDocumentCount	integer	No	Maximum number of documents to return	> 0, cannot be used with maxChunkCount
filters[].collectionMetadata	array	No	Metadata filters for collections	maxItems: 2000
filters[].collectionMetadata[].key	string	Yes (if parent used)	Metadata key to filter on	maxLength: 1024
filters[].collectionMetadata[].value	array of strings	Yes (if parent used)	Acceptable values for the key	maxLength per item: 1024
filters[].documentMetadata	array	No	Metadata filters for documents	maxItems: 2000
filters[].documentMetadata[].key	string	Yes (if parent used)	Document metadata key to filter on	maxLength: 1024
filters[].documentMetadata[].value	array of strings	Yes (if parent used)	Acceptable values for the key	maxLength per item: 1024
filters[].documentMetadata[].matchMode	string	No	Matching mode for document metadata values	"ANY" or "ALL"
filters[].documentMetadata[].selectMode	array	No	Selection mode options	N/A
filters[].documentMetadata[].selectMode[]	string	No	Selection mode option	"ignoreIfKeyAbsent"
filters[].chunkMetadata	array	No	Metadata filters for chunks	maxItems: 2000
filters[].chunkMetadata[].key	string	Yes (if parent used)	Chunk metadata key to filter on	maxLength: 1024
filters[].chunkMetadata[].value	array of strings	Yes (if parent used)	Acceptable values for the key	maxLength per item: 1024

1.4.3.2.1.1.2.4 Retrieval API

The Retrieval API lets you retrieve repositories or collections created through the Vector API. It also lets you perform similarity searches on the vector database to obtain relevant chunks and documents.

Prerequisites

You have created a vector database. For more information, see Vector API [page 477].

Capacity Quotas and Input Limits

Retrieval Search – Query Limits

The following query limits apply:

Category	Description	Limit Type	Limit
Filters	Max filters per query	Input Limit	15
Collections	Max collection IDs per query	Input Limit	5,000
Results	Max chunk results per query	Input Limit	100
Results	Max document results per query	Input Limit	10

API Request Format

When you build your API requests, follow the format outlined in the *Retrieval* section of the Grounding API. For more information, see Grounding API.

1.4.3.2.1.1.2.4.1 Retrieve a List of All Data Repositories

This endpoint allows you to retrieve a list of all repositories or vector collections stored in the vector database.

Procedure

You can retrieve the list by sending a GET request to the endpoint `$AI_API_URL/v2/lm/document-grounding/retrieval/dataRepositories`.

Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/retrieval/dataRepositories \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

Results

The response is returned in JSON format, and includes an array of data repository objects.

Pagination Support

You can use OData-compliant query parameters to manage large result sets:

Parameter	Type	Description
`$top`	Integer	Limits the number of results returned. For example, `$top=10` returns 10 results.
`$skip`	Integer	Skips a specific number of results. For example, `$skip=20` skips the first 20.
`$count`	Boolean	If `true`, includes the total count in the response. If `false`, includes the total number of records in the response.

Example

This code returns a JSON array of data repository objects.

```
curl --request GET
  --url "$AI_API_URL/v2/lm/document-grounding/retrieval/dataRepositories?
$top=10&$skip=20&$count=true"
  --header "AI-Resource-Group: {{resource_group}}"
  --header 'Authorization: Bearer {{access_token}}'
```

1.4.3.2.1.1.2.4.2 Retrieve a Specific Data Respository

Context

This endpoint allows you to retrieve a specific data repository or vector collection stored in the vector database.

Procedure

1. You can retrieve a data repository or vector collection by sending a GET request to the endpoint `$AI_API_URL/v2/lm/document-grounding/retrieval/dataRepositories/{{repositoryId}}`. Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core

Sample Code

```
curl --request GET \
  --url $AI_API_URL/v2/lm/document-grounding/retrieval/dataRepositories/
{{repositoryId}} \
  --header 'AI-Resource-Group: {{resource_group}}'
  --header 'Authorization: Bearer {{access_token}}'
```

This endpoint returns a response in JSON format. The response includes the specific data repository or vendor collection.

1.4.3.2.1.1.2.4.3 Retrieval Search

Context

This endpoint allows you to perform a similarity search between the user query and data repositories, returning relevant chunks that match the query.

Procedure

1. You can conduct this search by sending a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/retrieval/search`.
   Populate the sample code with the following values:

	Value
{{resource_group}}	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
{{access_token}}	Your access token for SAP AI Core
<user query>	The query that you want to be answered from the vector database
dataRepositories	`'*'` searches through all collections. Alternatively, enter a collection ID to retrieve a chunk from that collection.
dataRepositoryType	Supported repository types are `vector` and `help.sap.com`. Data from `help.sap.com` is available for retrieval from the database by default.

Examples

Example with SAP HANA Vector Store, using all “*” data repositories

Sample Code

```
curl --request POST \
--url $AI_API_URL/v2/lm/document-grounding/retrieval/search \
--header 'AI-Resource-Group: {{resource_group}}' \
--header 'Authorization: Bearer {{access_token}}'
--data '{
  "query": "<user query>",
  "filters": [
    {
      "id": "string",
      "searchConfiguration": {
        "maxChunkCount": 1
      },
      "dataRepositories": ['*']
      ],
      "dataRepositoryType": "vector",
      "dataRepositoryMetadata": [
      {
          "key": "type",
          "value": [
            "custom"
          ]
        }
      ],
      "documentMetadata": [
        {
```

```
          "key": "url",
          "value": [
            "http://hello.com",
            "123"
          ]
        }
      ],
      "chunkMetadata": [
        {
          "key": "index",
          "value": [
            "1"
          ]
        }
      ]
    }
  ]
}'
```

Example with help.sap.com

Sample Code

```
curl --request POST \
--url $AI_API_URL/v2/lm/document-grounding/retrieval/search \
--header 'AI-Resource-Group: {{resource_group}}' \
--header 'Authorization: Bearer {{access_token}}'
--data '{
    "query": "<user-query>",
    "filters": [
        {
        "id": "string",
        "searchConfiguration": {},
        "dataRepositories": ["*"],
        "dataRepositoryType": "help.sap.com",
        "dataRepositoryMetadata": [],
        "documentMetadata": []
    }'
}'
```

Results

This endpoint returns a response in JSON format. The response includes relevant chunk identified using the similarity search.

Chunk objects may also contain the following scores:

- `searchScores`: the relevance of each chunk to the search query, reported from the Vector Search. For more information see, Vector Search [page 498].
- `postProcessingScore`: the score after post processing the original score, for example with rerankers, Not all post processing operations will have a `postProcessingScore`.

Post Processing

Results from multiple repositories can be merged, by passing the results of the `filters` to the `postProcessing` field.

You define the `"inputs":` to specify which filter results will be included in post processing

> ⚠ Restriction
>
> Only Cohere 3.5 Reranker is supported.

Merging Strategies

The following strategies are supported for combining results from different filters:

- `scoreReuse`: Merges the scores of multiple filter-results by sorting their scores.
- `reranker`: Reorders the results of mulitple filters by using a reranker and their content.
  To include metadata in the reranking, include your metadata in the `"boosting"` field. Metadata is not used as a filter, but enhances the relevance of a chunk where a query is related to the metadata.

Example

The following example uses the reranker strategy and retrieval results from a vector similarity search.

> {·} Sample Code

```
{
  "query": "How much leave do I have?",
  "filters": [
    {
      "id_": "filter_1",
      "searchConfiguration": {
        "maxChunkCount": 20,
        "maxDocumentCount": null
      },
      "dataRepositories": [
        "*"
      ],
      "dataRepositoryType": "vector",
      "dataRepositoryMetadata": [
        {
          "key": "lob",
          "value": [
            "sfsf"
          ]
        }
      ],
      "documentMetadata": [
        {
          "key": "country",
          "value": [
            "US"
          ],
```

```
          "selectMode": null
        }
      ],
      "chunkMetadata": []
    }
  ],
  "postProcessing": [
    {
      "id_": "27b9aadb-7c9e-4f1a-86b5-ea419d2d4e67",
      "maxChunkCount": 5,
      "strategy": {
        "type_": "reranker",
        "model": "cohere-3.5",
        "boosting": [
          {
            "key": "country",
            "value": "US",
            "scope": "filter_1"
          }
        ],
        "includeMetaData": false
      },
      "inputs": [
        {
          "id_": "filter_1"
        }
      ]
    }
  ]
}
```

1.4.3.2.1.2 Using the Grounding Module

Prerequisites

You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].

Procedure

Populate the sample code with the following values:

Field	Value
$RESOURCE_GROUP	The AI resource group assigned to your account
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.

Field	Value
Bearer $TOKEN	Your access token for SAP AI Core
config.modules.grounding.type	`"document_grounding_service"`
config.modules.grounding.config.filters[].id	`"filter"`
config.modules.grounding.config.filters[].data_repositories	Array specifying which repositories to use for document grounding or "*" to use all available data repositories
config.modules.grounding.config.filters[].search_config.max_chunk_count	Integer that limits the maximum number of chunks to retrieve. Default: `10`
config.modules.grounding.config.filters[].data_repository_type	`"vector"`
config.modules.grounding.config.placeholders.input	A reference to the variable containing your grounding query
config.modules.grounding.config.placeholders.output	A variable that stores your grounding output
config.modules.prompt_templating.prompt.template	Your prompt template
config.modules.prompt_templating.model	Your model configuration
placeholder_values.groundingRequest	A string containing your grounding query

Examples

Example with SAP HANA Vector Store, using all “*” data repositories

Sample Code

```
curl --request POST "$ORCH_DEPLOYMENT_URL/v2/completion" \
--header "content-type: application/json" \
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP" \
--data-raw '{
    "config": {
        "modules": {
            "grounding": {
                "type": "document_grounding_service",
                "config": {
                    "filters": [
                        {
                            "id": "filter",
                            "data_repositories": [
                                "*"
                            ],
                            "search_config": {
                                "max_chunk_count": 10
                            },
                            "data_repository_type": "vector"
                        }
                    ],
                    "placeholders": {
                        "input": [
                            "groundingRequest"
                        ],
                        "output": "groundingOutput"
                    }
```

```
                }
            },
            "prompt_templating": {
                "prompt": {
                    "template": [
                        {
                            "role": "user",
                            "content": "<prompt>\n\nRequest: {{?
groundingRequest}}\nReports: {{?groundingOutput}}"
                        }
                    ]
                },
                "model": {
                    "name": "<modelName>",
                    "params": {
                        "max_completion_tokens": 200,
                        "temperature": 0.2
                    }
                }
            }
        }
    },
    "placeholder_values": {
        "groundingRequest": "<grounding query>"
    }
}'
```

Related Information

Leveraging Orchestration Capabilities to Enhance Responses
Libraries and SDKs

1.4.3.2.1.2.1 Contextualized Retrieval Using Metadata and Vector Search

You can choose to include metadata in the output structure of the grounding module in orchestration. You can leverage the metadata in your prompt template.

Prerequisites

You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].

You have onboarded a repository. For more information, see Generic Secrets for Grounding [page 398].

Context

Meta data inclusion is optional, and can be included using the `metadata_params` field in the completions endpoint of the API. The `metadata_params` field takes a list of comma separated strings.

Procedure

1. **Optional:** Retrieve supported metadata keys by sending your query to the endpoint `{{AI_API_Url}}/v2/lm/document-grounding/retrieval/search`.

   Sample Code

   ```
   curl --request POST \
     --url {{AI_API_URL}}/v2/lm/document-grounding/retrieval/search \
     --header 'AI-Resource-Group: <resource_group>' \
     --header 'Authorization: Bearer <access_token>' \
     --header 'Content-Type: application/json' \
     --data '{
       "query": "what is SAP Joule",
       "filters": [
           {
               "id": "string",
               "searchConfiguration": {
                   "maxChunkCount": 5
               },
               "dataRepositories": [
                   "*"
               ],
               "dataRepositoryType": "vector",
               "dataRepositoryMetadata": [
                   {
                       "key": "type",
                       "value": [
                           "custom"
                       ]
                   }
               ],
               "documentMetadata": [],
               "chunkMetadata": []
           }
       ]
   }'
   ```

   The response includes details of metadata parameters on the following levels: `dataRepository`, `document` and `chunk`.

2. To include metadata as part of your request, use `metadata_params` and include them as comma separated strings.

   The following example requests that `source` and `webUrl` metadata is included in the results

   Sample Code

   ```
   curl --request POST \
     --url {{ORCH_DEPLOYMENT_URL}}/v2/completion \
     --header 'Authorization: Bearer <access_token>' \
     --header 'AI-Resource-Group: <resource_group>' \
   ```

```
  --header 'content-type: application/json' \
  --data '{
    "config": {
        "modules": {
            "grounding": {
                "type": "document_grounding_service",
                "config": {
                    "filters": [
                        {
                            "id": "filter1",
                            "data_repositories": [
                                "*"
                            ],
                            "search_config": {
                                "max_chunk_count": 100
                            },
                            "data_repository_type": "vector"
                        }
                    ],
                    "placeholders": {
                        "input": [
                            "groundingRequest"
                        ],
                        "output": "groundingOutput"
                    },
                    "metadata_params": [
                        "source",
                        "webUrl"
                    ]
                }
            },
            "prompt_templating": {
                "prompt": {
                    "template": [
                        {
                            "role": "user",
                            "content": "<prompt>: {{?groundingRequest}}
\n\nReports: {{?groundingOutput}}"
                        }
                    ]
                },
                "model": {
                    "name": "<modelName>",
                    "version": "<modelVersion>",
                    "params": {
                        "max_completion_tokens": 50,
                        "temperature": 0.1,
                        "frequency_penalty": 0,
                        "presence_penalty": 0
                    }
                }
            }
        }
    },
    "placeholder_values": {
        "groundingRequest": "what is SAP Joule?"
    }
}'
```

Output Code

```
[
    [
        {
```

```
            "content": "Joule is the AI copilot that truly understands
your business. Joule revolutionizes how you interact with your SAP
business systems, making every touchpoint count and every task simpler.",
            "metadata": {
                "source": [
                    "/tmp/document_1.pdf"
                ],
                "webUrl": [
                    "https://<sharepoint_url>/sites/<Documents>/
sap_joule.pdf"
                ],
                "document_webUrl": [
                    "https://<sharepoint_url>/sites/<Documents>/
sap_joule.pdf"
                ]
            }
        },
        {
            "content": "AI-powered capabilities in Joule with custom-
developed enterprise skills tailored to your company's specific needs.
By integrating with both SAP and third-party applications, Joule can
instantly apply its newfound knowledge within SAP systems.",
            "metadata": {
                "source": [
                    "/tmp/document_1.pdf"
                ],
                "webUrl": [
                    "https://<sharepoint_url>/sites/<Documents>/
sap_joule.pdf"
                ],
                "document_webUrl": [
                    "https://<sharepoint_url>/sites/<Documents>/
sap_joule.pdf"
                ]
            }
        }
    ]
]
```

3. Leverage the metadata in your prompt

   Include instructions in your prompt for how the model should use the metadata.

{·} Sample Code

```
"prompt_templating": {
  "messages_history": [
    {
      "role": "user",
      "content": "You are a helpful assistant for any queries for
SAP Teched 2024.\\nAnswer the grounding request by providing relevant
answers that fit to the request. \\n\\nRequest: {{ ?groundingRequest }}\\n\
\nReports:{{ ?groundingOutput }} Alongwith the original response, provide
another response by making use of metadata received in the grounding
output to provide more optimal response. Explain how the metadata was
used."
     }
      ],
      "defaults": {}
  }
```

ⓘ Note

If a metadata name is present on multiple levels, the result will include a `parent_KeyName` naming convention. The following example shows that `webUrl` is present on the chunk and document levels:

Output Code

```
"metadata": {
  "webUrl": "<value from chunk metadata>",
  "document_webUrl": "<value from document metadata>"
}
```

1.4.3.2.1.2.2 Retrieval Using SAP Help Portal

Prerequisites

- You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].
- You've onboarded a repository. For more information, see Generic Secrets for Grounding [page 398].

Context

Meta data inclusion is optional, and can be included using the `metadata_params` field in the completions endpoint of the API. The `metadata_params` field takes a list of comma-separated strings.

Procedure

1. **Optional:** Retrieve supported metadata keys by sending your query to the endpoint `{{AI_API_Url}}/v2/lm/document-grounding/retrieval/search`.

   Sample Code

   ```
   curl --request POST \
     --url "$AI_API_URL/v2/lm/document-grounding/retrieval/search" \
     --header "AI-Resource-Group: <resource_group>" \
     --header "Authorization: Bearer <access_token>" \
     --header "Content-Type: application/json" \
     --data '{
       "query": "what is SAP Joule",
       "filters": [
         {
           "id": "string",
           "searchConfiguration": {},
           "dataRepositories": []
         }
       ],
   ```

```
    "documentMetadata": [],
    "chunkMetadata": []
  }'
```

The response includes details of metadata parameters on the following levels: `dataRepository`, `document`, and `chunk`.

2. To include metadata as part of your request, use `metadata_params` and include them as comma-separated strings.

   The following example requests that `source` and `webUrl` metadata is included in the results

{·} Sample Code

```
curl --request POST \
  --url "$ORCH_DEPLOYMENT_URL/v2/completion" \
  --header "Authorization: Bearer <access_token>" \
  --header "AI-Resource-Group: <resource_group>" \
  --header "Content-Type: application/json" \
  --data '{
    "config": {
      "modules": {
        "grounding": {
          "type": "document_grounding_service",
          "config": {
            "filters": [
              {
                "id": "filter1",
                "data_repositories": ["*"],
                "search_config": {},
                "data_repository_type": "help.sap.com",
                "document_metadata": []
              }
            ]
          }
        }
      },
      "placeholders": {
        "input": ["groundingRequest"],
        "output": "groundingOutput",
        "prompt_templating": {
          "prompt": {
            "template": [
              {
                "role": "user",
                "content": "<prompt>: {{?groundingRequest}}\n\nReports: {{?
groundingOutput}}"
              }
            ]
          },
          "model": {
            "name": "<modelName>",
            "version": "<modelVersion>",
            "params": {
              "max_completion_tokens": 50,
              "temperature": 0.1,
              "frequency_penalty": 0,
              "presence_penalty": 0
            }
          }
        }
      }
    },
    "placeholder_values": {
      "groundingRequest": "<grounding request>"
    }
```

```
}'
```

1.4.3.2.2 Templating

The templating module is mandatory. It enables you to:

- Compose prompts
- Define placeholders
- Configure your chosen LLM

Your template generates the final query and inferences your specified LLM.

File formats are supported based on the capabilities of the model provider.

> **Note**
>
> **Caching**
>
> Implicit context caching using Gemini and Azure OpenAI models is enabled by default in Orchestration.
>
> Explicit context caching, including Amazon Bedrock and Anthropic models, is not supported in Orchestration.

You can configure the module by passing the following information:

- Model Name (parameter `name`): The model name is required. For information about foundation models and supported models, see Foundation Models [page 190] and SAP Note 3437766.
- Model Parameters (parameter `params`): The model parameters are optional. Possible values depend on the chosen model. For more information, see Harmonized API [page 570].
  For Anthropic models, the `max_completion_tokens` parameter is required. If you don't set a value, the orchestration service sets this parameter to the maximum value allowed by the model. For more information, see the Anthropic documentation at Models .
- Model Version (parameter `version`): The model version is optional and defaults to “latest”.
- Timeout (parameter `timeout`): is optional and is specified in seconds. Min: 1, max: 1200, default: 600.
  You can use timeout to configure a timeout limit for LLM calls.
  Orchestration will either retry or fail the request with a read timeout error (408) if the timeout elapses.
  For streaming requests, the timeout represents the entire duration of the streaming response, from connection establishment until the stream is either completely consumed or closed.
  This parameter is ignored by Vertex AI models.
- Max retries (parameter `max_retries`): is optional and is specified in number of retries. Min: 0, max: 5, default: 2.
  You can use max_retries to configure a number of retries for LLM calls. Retries are attempted in case of errors related to connection, network and read timeouts, for example.
  For streaming requests, retries are only attempted when establishing the initial connection. Retries are not attempted after streaming responses have started.
  This parameter is ignored by Vertex AI models.

Any placeholders that you define in your prompt can be filled at runtime. For example, the following template places the input text into the `{{ ?input }}` placeholder:

```
 "prompt_templating": {
         "prompt": {
           "template": [
           {
             "role": "user",
             "content": "{{?input}}"
           }
         ]
         },
         "model": {
           "name": "gpt-4o",
           "params": {
             "max_completion_tokens": 300,
             "temperature": 2
           }
           "timeout": 10     // in seconds
           "max_retries": 3
        }
       }
```

The `{{ ?input }}` placeholder is filled using the contents of the `input_params.input` parameter:

```
"input_params": {
  "input": "A sample prompt to be sent to the model"
}
```

You can also set default values for the placeholders. For example, to request either 5 paraphrases or a user-specific number of paraphrases for a given phrase, you can use the following configuration:

```
{
  "prompt_templating": {
    "prompt": {
      "template": [
        {
          "role": "user",
          "content": "Create {{ ?number }} paraphrases of {{ ?phrase }}"
        }
      ],
      "defaults": {
        "number": 5
      }
    }
  },
  //... other configuration
    "placeholder_values": {
      "number": "3",
      "phrase": "Hello."
    }
}
```

Templates can also be retrieved from the prompt registry. For more information, see Prompt Registry [page 732].

Templates can be retrieved by ID (immutable) or the combination of scenario, name, and version (mutable). Additionally, a 'scope' parameter can be specified to define whether the requested template is shared across

all resource groups within the 'tenant' (default), or is only accessible within the specific 'resource_group'. For example:

Sample Code

```
"prompt_templating": {
          "prompt": {
              "template_ref": {
                  "id": "d22342e7-1f91-4c52-a794-04833bc2574a"
                  "scope": "tenant"
          }
    }
},
```

or

```
"prompt_templating": {
          "prompt": {
              "template_ref": {
                  "scenario": "test-scenario",
                  "name": "test-template",
                  "version": "0.0.1"
                  "scope": "tenant"
              }
          }
 },
```

Input Image Support

Images can be provided as additional input where supported.

For example:

Sample Code

```
  "messages_history": [
     {"role": "system", "content": "You are a helpful assistant."},
     {
         "role": "user",
         "content": [
             {"type": "text", "text": "what is in this image?"},
             {"type": "image_url", "image_url": {"url": "..."}},
             ],
         },

 ],
```

The URL for `image_url` supports Base64 or a web-based URL of the PDF. Check with the model provider that your preferred upload method is supported.

```
"image_url": {
    "url":  f"{base64_image}"
}
```

or

```
 "image_url": {
```

```
    "url": "https://some.jpg"
}
```

Input PDF Support

PDFs can be provided as additional input where supported by the model provider.

```
{
  "prompt_templating": {
    "prompt": {
      "template": [
        {
          "role": "system",
          "content": "You describe files. You keep it very short."
        },
        {
          "role": "user",
          "content": [
            {
              "type": "text",
              "text": "Summarize the document."
            },
            {
              "type": "file",
              "file": {
                "file_data": "data:application/pdf; base64,{{ ?
base64_green_square }}"
              }
            }
          ]
        }
      ]
    }
  }
}
```

The `file_data` field supports either Base64-encoded content or a web-based URL. Verify with the model provider that your preferred upload method is supported.

For example:

```
"file_data": "data:application/pdf; base64,{{ ?base64_green_square }}"
```

or

```
 "file_data": "https://somefile.pdf"
```

Related Information

Access the Templating using the SAP AI SDK for Java

Access the Templating using the SAP AI SDK for JavaScript

1.4.3.2.2.1 Prompt Caching

Prompt caching improves performance by reusing predefined sections of a prompt across multiple orchestration requests.

Orchestration supports explicit prompt caching by defining `cache_control` breakpoints on content blocks in the `prompt_templating` module of a request.

Explicit prompt caching is available only in Orchestration V2.

> **Note**
>
> Implicit context caching for OpenAI and Gemini models is enabled by default in orchestration and does not require additional configuration.

Supported Models

Explicit prompt caching using `cache_control` is supported for the following model families:

- **Anthropic Claude models** support `cache_control` on `system`, `messages`, and `tools` content blocks. A single request supports up to four cache breakpoints. Cache entries are created in the following order: tools, system, and messages.
- **Amazon Nova models** support `cache_control` on `system` and `messages` content blocks. A single request typically supports one cache breakpoint.

Limitations

Explicit prompt caching using cache_control is supported only for Anthropic Claude models and Amazon Nova models.

Amazon Nova models do not support cache_control on tools content blocks.

The one-hour time-to-live value is supported only by select Anthropic models.

Prompt caching requires a minimum number of tokens in the prompt prefix. If the prompt is too short, the cache breakpoint may not take effect. For more information, see the Anthropic prompt caching documentation at https://docs.anthropic.com/en/docs/build-with-claude/prompt-caching and the Amazon Bedrock prompt caching documentation at https://docs.aws.amazon.com/bedrock/latest/userguide/prompt-caching.html .

Cache TTL (Time-to-Live)

By default, cached content uses a five-minute TTL.

```
"cache_control": {"type": "ephemeral"}
```

For selected Anthropic Claude models, you can optionally configure a one-hour TTL by setting the `ttl` field.

```
"cache_control": {"type": "ephemeral", "ttl": "1h"}
```

Request Examples

Send a POST request to the following endpoint: **{{Orchestration URL}}/v2/completion**

Cache Control on System and Message Content Blocks

The following example shows `cache_control` breakpoints defined on a system message and a user message. This approach is useful when caching long system prompts together with multi-turn few-shot examples.

```
{
  "config": {
    "modules": {
      "prompt_templating": {
        "prompt": {
          "template": [
            {
              "role": "system",
              "content": [
                {
                  "type": "text",
                  "text": "You are an expert news article classifier. Classify
each article into exactly one category.",
                  "cache_control": { "type": "ephemeral" }
                }
              ]
            },
            {
              "role": "user",
              "content": "input: Comcast launches prepaid plans"
            },
            {
              "role": "assistant",
              "content": "Business"
            },
            {
              "role": "user",
              "content": "input: Fed signals rate cuts may come later than
markets expected"
            },
            {
              "role": "assistant",
              "content": "Economics"
            },
            {
              "role": "user",
              "content": [
                {
                  "type": "text",
                  "text": "input: {{ ?input }}",
                  "cache_control": { "type": "ephemeral" }
```

```
                }
              ]
            }
          ]
        },
        "model": {
          "name": "anthropic--claude-4-opus",
          "params": {
            "max_tokens": 50,
            "temperature": 0.1
          }
        }
      }
    }
  },
  "placeholder_values": {
    "input": "Scaling up neural models has yielded significant advancements in
language generation"
  }
}
```

Cache Control on Tools (Anthropic Claude Only)

You can also apply `cache_control` to tool definitions. Add the `cache_control` field at the same level as `type` and `function`.

```
"tools": [
  {
    "type": "function",
    "function": {
      "name": "classify_article",
      "description": "Classify a news article into a predefined category.",
      "parameters": {
        "type": "object",
        "properties": {
          "category": {
            "type": "string",
            "enum": ["Business", "Economics", "Tech"],
            "description": "The primary category of the article"
          }
        },
        "required": ["category"]
      }
    },
    "cache_control": { "type": "ephemeral" }
  }
]
```

Amazon Nova Models

For Amazon Nova models, use the same `cache_control` syntax. The orchestration service automatically converts `cache_control` to the native Nova cachePoint format.

```
"model": {
```

```
  "name": "amazon--nova-pro",
  "params": {
    "max_tokens": 200
  }
}
```

> **Note**
>
> Amazon Nova models support cache control only on system and messages content blocks. Cache control on tools and explicit TTL values is not supported.

Response

When prompt caching is enabled, the response usage object includes additional fields in `prompt_tokens_details` to report cache usage.

Response with Cache Usage Using Default Time to Live

When cache_control is used without an explicit time-to-live value, the response includes both cached_tokens and cache_creation_tokens in the prompt_tokens_details object.

```
{
  "usage": {
    "completion_tokens": 4,
    "prompt_tokens": 3,
    "total_tokens": 7,
    "prompt_tokens_details": {
      "cached_tokens": 4657,
      "cache_creation_tokens": 0
    }
  }
}
```

`cached_tokens` indicates the number of input tokens read from the cache.

`cache_creation_tokens` indicates the number of input tokens written to the cache.

Response with Cache Usage Using Explicit Time to Live

When cache_control includes an explicit time-to-live value, the response adds cache_creation_token_details to show a breakdown of cached tokens by duration.

```
{
  "usage": {
    "completion_tokens": 4,
    "prompt_tokens": 3,
    "total_tokens": 7,
```

```
    "prompt_tokens_details": {
      "cached_tokens": 4657,
      "cache_creation_tokens": 0,
      "cache_creation_token_details": {
        "ephemeral_5m_input_tokens": 0,
        "ephemeral_1h_input_tokens": 0
      }
    }
  }
}
```

`ephemeral_5m_input_tokens` represents tokens cached with a five-minute duration.

`ephemeral_1h_input_tokens` represents tokens cached with a one-hour duration.

1.4.3.2.2.2 Few-Shot Learning

Prerequisites

You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].

Process

The following example shows how you can configure the templating module to use a few-shot learning prompt.

```
curl --request POST $ORCH_DEPLOYMENT_URL/v2/completion \
    --header 'content-type: application/json' \
    --header "Authorization: Bearer $TOKEN" \
    --header "AI-Resource-Group: $RESOURCE_GROUP" \
    --data-raw '{
    "config": {
      "modules": {
        "prompt_templating": {
          "prompt": {
                "template": [
                  {
                    "role": "system",
                    "content": "You classify input text into the two following
categories: Business, Economics, Tech, and other"
                  },
                  {
                    "role": "user",
                    "content": "input text: `Comcast launches prepaid plans`"
                  },
                  {
                    "role": "assistant",
                    "content": "Business"
                  },
                  {
```

```
                    "role": "user",
                    "content": "input text: `Slower Fed Pivot Weakens Rate-Cut
Bets Across Emerging Asia`"
                  },
                  {
                    "role": "assistant",
                    "content": "Economics"
                  },
                  {
                    "role": "user",
                    "content": "input text: {{?input}}"
                  }
              ]
          },
          "model": {
            "name": "gpt-4o",
            "params": {
              "max_completion_tokens": 300,
              "temperature": 2
            }
          }
        }
      }
    },
    "placeholder_values": {
        "input": "Scaling up neural models has yielded significant advancements
in a wide array of tasks, particularly in language generation."
    }
}
'
```

In this case, the template contains an array of messages, including a system message, as well as several user and assistant messages. The actual input to categories is configured with the final user message and the content `input text: {{?input}}`. The input parameter input is set to `Scaling up neural models has yielded significant advancements in a wide array of tasks, particularly in language generation.`.

Related Information

Leveraging Orchestration Capabilities to Enhance Responses

Libraries and SDKs

1.4.3.2.2.3 Chat

Orchestration can also be used in chat scenarios. The following example shows how to configure the templating module to use a chat prompt.

Prerequisites

You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].

Process

```
curl --request POST $ORCH_DEPLOYMENT_URL/v2/completion \
    --header 'content-type: application/json' \
    --header "Authorization: Bearer $TOKEN" \
    --header "AI-Resource-Group: $RESOURCE_GROUP" \
    --data-raw '{
    "config": {
      "modules": {
        "prompt_templating": {
          "prompt": {
            "template": [
              {
                "role": "user",
                "content": "And now two more that sound like Goethe."
              }
            ]
          },
          "model": {
            "name": "gpt-4o-mini",
            "params": {
              "max_completion_tokens": 300,
              "temperature": 0.1,
              "frequency_penalty": 0,
              "presence_penalty": 0
            }
          }
        }
      }
    },
    "placeholder_values": {
    },
    "messages_history": [
      {
        "role":"user",
        "content":"Create 3 paraphrases of I love you."
      },
      {
        "role":"assistant",
        "content":"1. I have deep affection for you.\n2. My feelings for you are
strong and caring.\n3. You mean everything to me."
      },
      {
        "role":"user",
```

```
        "content":"Can you please make them more romantic."
      },
      {
        "role":"assistant",
        "content":"1. My heart is filled with an intense passion for you.\n2.
You are the light of my life, and my love for you knows no bounds.\n3. I cherish
you more than words can express, my dear."
      },
      {
        "role":"user",
        "content":"Try sounding like shakespeare."
      },
      {
        "role":"assistant",
        "content":"1. Thou art the sun in my sky, and my love for thee doth
burn with unyielding fervor.\n2. Within mine heart, thou hast ignited a flame of
ardor that doth shine brightly for thee, my dearest.\n3. My love for thee doth
surpass all measure, for thou art the true essence of my being."
      }
    ]
  }'
```

As shown above, messages from the chat can be passed to the current request using the `messages_history` parameter. In this case, the chat history contains several user messages and assistant responses. The templating module is then configured with the current user message, which is used to generate the next assistant response. The templating module appends the current user message to the message history to generate the prompt that is sent to the LLM module.

The response contains the messages from the chat history and the response to the new message. The output of `intermediate_results.templating` and the `final_result.choices` results can be used as the message history for a subsequent inference request:

```
{
  "request_id": "9a66158b-b019-951a-a5e7-3c3ec9d2662e",
  "intermediate_results": {
    "templating": [
      {
        "content": "Create 3 paraphrases of I love you.",
        "role": "user"
      },
      {
        "role": "assistant",
        "content": "1. I have deep affection for you.\n2. My feelings for you
are strong and caring.\n3. You mean everything to me."
      },
      {
        "content": "Can you please make them more romantic.",
        "role": "user"
      },
      {
        "role": "assistant",
        "content": "1. My heart is filled with an intense passion for you.\n2.
You are the light of my life, and my love for you knows no bounds.\n3. I cherish
you more than words can express, my dear."
      },
      {
        "content": "Try sounding like shakespeare.",
        "role": "user"
      },
      {
        "role": "assistant",
        "content": "1. Thou art the sun in my sky, and my love for thee doth
burn with unyielding fervor.\n2. Within mine heart, thou hast ignited a flame of
ardor that doth shine brightly for thee, my dearest.\n3. My love for thee doth
surpass all measure, for thou art the true essence of my being."
```

```
      },
      {
        "content": "And now two more that sound like Goethe.",
        "role": "user"
      }
    ],
    "llm": {
      "id": "chatcmpl-BuHIHDDcx0FPsOMVEXYp8eVKA4ulw",
      "object": "chat.completion",
      "created": 1752752545,
      "model": "gpt-4o-mini-2024-07-18",
      "system_fingerprint": "fp_efad92c60b",
      "choices": [
        {
          "index": 0,
          "message": {
            "role": "assistant",
            "content": "1. In the depths of my soul, I find an eternal longing
for thee, a love that transcends the bounds of time and space.\n2. Thou art the
muse of my existence, inspiring a passion within me that resonates with the very
essence of nature itself."
          },
          "finish_reason": "stop"
        }
      ],
      "usage": {
        "completion_tokens": 55,
        "prompt_tokens": 212,
        "total_tokens": 267
      }
    }
  },
  "final_result": {
    "id": "chatcmpl-BuHIHDDcx0FPsOMVEXYp8eVKA4ulw",
    "object": "chat.completion",
    "created": 1752752545,
    "model": "gpt-4o-mini-2024-07-18",
    "system_fingerprint": "fp_efad92c60b",
    "choices": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": "1. In the depths of my soul, I find an eternal longing
for thee, a love that transcends the bounds of time and space.\n2. Thou art the
muse of my existence, inspiring a passion within me that resonates with the very
essence of nature itself."
        },
        "finish_reason": "stop"
      }
    ],
    "usage": {
      "completion_tokens": 55,
      "prompt_tokens": 212,
      "total_tokens": 267
    }
  }
}
```


> **Note**
>
> When other orchestration modules are configured alongside chat, `messages_history` is included in the orchestration pipeline. The templating module prepends `messages_history` to the current template

before passing the combined conversation to downstream modules. This impacts the following points in the workflow:

- **Data masking:** All messages, including those from `messages_history`, are masked before reaching the model.
- **Content filtering:** All messages, including those from `messages_history`, are evaluated by the configured content filters.
- **Translation:** By default, `messages_history` is translated together with the template messages. To skip translation of the history, set `translate_messages_history` to `false` in the input translation configuration.

Related Information

Leveraging Orchestration Capabilities to Enhance Responses
Libraries and SDKs

1.4.3.2.2.4 Minimal Call

Prerequisites

You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].

Process

A minimal call to orchestration contains only configurations of the required templating and model configuration modules. The curl command below shows how to make such a request.

```
curl --request POST $ORCH_DEPLOYMENT_URL/v2/completion \
    --header 'content-type: application/json' \
    --header "Authorization: Bearer $TOKEN" \
    --header "AI-Resource-Group: $RESOURCE_GROUP" \
    --data-raw '{
    "config": {
      "modules": {
        "prompt_templating": {
          "prompt": {
            "template": [
            {
              "role": "user",
```

```
              "content": "Reply with `{{?text}}` in {{?language}}"
            }
          ]
          },
          "model": {
            "name": "gpt-4o-mini",
            "params": {
              "max_completions_tokens": 300,
              "temperature": 2
            }
        }
      }
    }
  },
  "placeholder_values": {
    "text": "Hello",
    "language": "German"
  }
}
'
```

This request configures the templating module with a single user message with two parameters: `text` and `language`. The `language` parameter is also configured with English as the default. The LLM module is configured to use gpt-4o-mini in the latest available version and a set of model parameters. The `placeholder_values` field contains the values for the parameters `text` and `language`. These values are used during this request in the prompt sent to the model.

The response contains a `request_id`, the module results from each module that was executed, and the `orchestration_result`, which includes the response of the call to the model.

Output Code


```
{
  "request_id": "befbe737-5e2c-96fd-a30b-aa18c500a316",
  "intermediate_results": {
    "templating": [
      {
        "content": "Reply with `Hello` in German",
        "role": "user"
      }
    ],
    "llm": {
      "id": "chatcmpl-BuHKkcuKYRrriwYdOYTzAol15mwya",
      "object": "chat.completion",
      "created": 1752752698,
      "model": "gpt-4o-mini-2024-07-18",
      "system_fingerprint": "fp_efad92c60b",
      "choices": [
        {
          "index": 0,
          "message": {
            "role": "assistant",
            "content": "Hallo!"
          },
          "finish_reason": "stop"
        }
      ],
      "usage": {
        "completion_tokens": 3,
        "prompt_tokens": 14,
        "total_tokens": 17
      }
    }
  },
  "final_result": {
```

```
    "id": "chatcmpl-BuHKkcuKYRrriwYdOYTzAol15mwya",
    "object": "chat.completion",
    "created": 1752752698,
    "model": "gpt-4o-mini-2024-07-18",
    "system_fingerprint": "fp_efad92c60b",
    "choices": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": "Hallo!"
        },
        "finish_reason": "stop"
      }
    ],
    "usage": {
      "completion_tokens": 3,
      "prompt_tokens": 14,
      "total_tokens": 17
    }
  }
}
```


The templating module result contains the user message with the filled in parameters. The LLM module result contains the response of the model execution. In this example, the LLM module result and the orchestration result are the same. However, they might differ, such as when the output filtering module filters the response.

Related Information

Leveraging Orchestration Capabilities to Enhance Responses
Libraries and SDKs

1.4.3.2.2.5 Tool Calling

You can incorporate specialist tools into your orchestration workflow for example to make an API call for current data or to execute custom code. LLMs decide that a tool call is needed based on your system prompt, messages and tool definition. You then execute your function code, return the results and the model will incorporate the results into its response.

You can define multiple tools, and use the decision making capabilities of generative AI models to guide you in their use. Tools only get called when the model determines that they are needed, based on your input and tool definitions.

Example

The following code shows a tool containing a custom function for an API call to retrieve item quantity information from an inventory:

Sample Code

```
import requests
def get_inventory_quantity(product_id):
    response = requests.get(f"https://my-store-api.com/v1/inventory?
product_id={product_id}")
    data = response.json()
    return data['product_id']['quantity']
```

You can include tools in your template by providing and outline of the function and its arguments in your template. You can include guidance on when a tool should be used in your system messages.

The following call includes an outline of the get_inventory_quantity function above in the `tools` parameter, with a description of its use case in the system role content. The `placeholder_values` include a request for product ID 123456.

Sample Code

```
curl --request POST \
  --url {{orchestration_deployment_url}}/v2/completion \
  --header 'AI-Resource-Group: default' \
  --header 'Authorization: Bearer {{token}}' \
  --header 'content-type: application/json' \
  --data '{
  "config": {
    "modules": {
      "prompt_templating": {
        "prompt": {
          "template": [
          {
            "role": "system",
            "content": "You are a helpful assistant. You help users to
analyze inventory of the user's store by calling the get_inventory_quantity
function."
          },
          {
            "role": "user",
            "content": "How many units of the product with ID {{ ?
product_id }} are in stock?"
          }
        ],
        "tools": [
          {
            "type": "function",
            "function": {
              "name": "get_inventory_quantity",
              "description": "Check available quantity for a product ID.",
              "strict": true,
              "parameters": {
                "type": "object",
                "properties": {
                  "product_id": {
                    "type": "integer"
                  }
                },
                "required": [
                  "product_id"
```

```
                ],
                "additionalProperties": false
              }
            }
          }
        ]
        },
        "model": {
          "name": "gpt-4o",
          "params": {
            "max_completion_tokens": 200
          }
        }
      }
    }
  },
  "placeholder_values": {
    "product_id": "123456"
  }
}'
```

The model uses the tool definition to decide that a function call is needed, and returns the function name and its input arguments in the format defined in the template. Use the information provided to call your custom function.

Output Code

```
{
  "request_id": "c3a3aeb5-5044-9152-8c3b-cf2fa4c07b25",
  "intermediate_results": {
    "templating": [
      {
        "role": "system",
        "content": "You are a helpful assistant. You help users to analyze
inventory of the user's store by calling the get_inventory_quantity function."
      },
      {
        "content": "How many units of the product with ID 123456 are in
stock?",
        "role": "user"
      }
    ],
    "llm": {
      "id": "chatcmpl-BuZqH3BWFuVVuge7Hg73ecLutnTje",
      "object": "chat.completion",
      "created": 1752823845,
      "model": "gpt-4o-2024-08-06",
      "system_fingerprint": "fp_ee1d74bde0",
      "choices": [
        {
          "index": 0,
          "message": {
            "role": "assistant",
            "content": "",
            "tool_calls": [
              {
                "id": "call_9yP43Z8VwoEQoB8mCdfsUQ3b",
                "type": "function",
                "function": {
                  "name": "get_inventory_quantity",
                  "arguments": "{\"product_id\":123456}"
                }
              }
            ]
          },
```

```
          "finish_reason": "tool_calls"
        }
      ],
      "usage": {
        "completion_tokens": 18,
        "prompt_tokens": 86,
        "total_tokens": 104
      }
    }
  },
  "final_result": {
    "id": "chatcmpl-BuZqH3BWFuVVuge7Hg73ecLutnTje",
    "object": "chat.completion",
    "created": 1752823845,
    "model": "gpt-4o-2024-08-06",
    "system_fingerprint": "fp_ee1d74bde0",
    "choices": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": "",
          "tool_calls": [
            {
              "id": "call_9yP43Z8VwoEQoB8mCdfsUQ3b",
              "type": "function",
              "function": {
                "name": "get_inventory_quantity",
                "arguments": "{\"product_id\":123456}"
              }
            }
          ]
        },
        "finish_reason": "tool_calls"
      }
    ],
    "usage": {
      "completion_tokens": 18,
      "prompt_tokens": 86,
      "total_tokens": 104
    }
  }
}
```


When you have called your custom function, use the response to extend your LLM call.

In the following example, the function call has returned a quantity of 25 items in integer format. This information is inserted into the `template` in the `tool` role.

Sample Code

```
curl --request POST \
  --url {{orchestration_deployment_url}}/v2/completion \
  --header 'AI-Resource-Group: default' \
  --header 'Authorization: Bearer {{token}}' \
  --header 'content-type: application/json' \
  --data '{
  "config": {
    "modules": {
      "prompt_templating": {
        "prompt": {
          "template": [
          {
            "role": "system",
```

```
            "content": "You are a helpful assistant. You help
users to analyze inventory of the user'\''s store by calling the
get_inventory_quantity function."
          },
          {
            "role": "user",
            "content": "How many units of the product with ID {{ ?
product_id }} are in stock?"
          },
          # append the function call message
          {
            "role": "assistant",
            "content": "",
            "tool_calls": [
              {
                "id": "call_il3KDaSC5zm6naTOnYv5VSZT",
                "type": "function",
                "function": {
                  "name": "get_inventory_quantity",
                  "arguments": "{\"product_id\":123456}"
                }
              }
            ]
          },
          # append function result
          {
            "role": "tool",
            "tool_call_id": "call_il3KDaSC5zm6naTOnYv5VSZT",
            "content": "25"
          }
        ],
        "tools": [
          {
            "type": "function",
            "function": {
              "name": "get_inventory_quantity",
              "description": "Check available quantity for a product ID.",
              "strict": true,
              "parameters": {
                "type": "object",
                "properties": {
                  "product_id": {
                    "type": "integer"
                  }
                },
                "required": [
                  "product_id"
                ],
                "additionalProperties": false
              }
            }
          }
        ]
        },
        "model": {
          "name": "gpt-4o",
          "params": {
            "max_completion_tokens": 200
          }
        }
      }
    }
  },
  "placeholder_values": {
    "product_id": "123456"
  }
}'
```

The model answers the query, incorporating the result of the function call into its response:

Output Code


```
{
  "request_id": "9b34a47b-14f1-95d3-9f84-51bdc4d66c2d",
  "intermediate_results": {
    "templating": [
      {
        "role": "system",
        "content": "You are a helpful assistant. You help users to analyze
inventory of the users store by calling the get_inventory_quantity function."
      },
      {
        "content": "How many units of the product with ID 123456 are in
stock?",
        "role": "user"
      },
      {
        "role": "assistant",
        "content": "",
        "tool_calls": [
          {
            "id": "call_il3KDaSC5zm6naTOnYv5VSZT",
            "type": "function",
            "function": {
              "name": "get_inventory_quantity",
              "arguments": "{\"product_id\":123456}"
            }
          }
        ]
      },
      {
        "role": "tool",
        "tool_call_id": "call_il3KDaSC5zm6naTOnYv5VSZT",
        "content": "25"
      }
    ],
    "llm": {
      "id": "chatcmpl-BuZvTwr6ZSxebgqHn8vp46zntVcZ1",
      "object": "chat.completion",
      "created": 1752824167,
      "model": "gpt-4o-2024-08-06",
      "system_fingerprint": "fp_ee1d74bde0",
      "choices": [
        {
          "index": 0,
          "message": {
            "role": "assistant",
            "content": "There are 25 units of the product with ID 123456 in
stock."
          },
          "finish_reason": "stop"
        }
      ],
      "usage": {
        "completion_tokens": 18,
        "prompt_tokens": 113,
        "total_tokens": 131
      }
    }
  },
  "final_result": {
    "id": "chatcmpl-BuZvTwr6ZSxebgqHn8vp46zntVcZ1",
    "object": "chat.completion",
    "created": 1752824167,
    "model": "gpt-4o-2024-08-06",
```

```
    "system_fingerprint": "fp_ee1d74bde0",
    "choices": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": "There are 25 units of the product with ID 123456 in
stock."
        },
        "finish_reason": "stop"
      }
    ],
    "usage": {
      "completion_tokens": 18,
      "prompt_tokens": 113,
      "total_tokens": 131
    }
  }
}
```

Related Information

Access the Tool Calling using the SAP AI SDK for Java

Access the Tool Calling using the SAP AI SDK for JavaScript

Access Tool Calling for Azure OpenAI

Access Tool Calling for OpenAI

1.4.3.2.2.5.1 Tool Calling with Other Orchestration Modules

> → Tip
>
> For use cases including both prompt optimization and tool calling, the `FC:ExactMatch` metric is recommended. Other metrics may return results that are unreliable.

Data Masking (Pseudonymization)

When tool calling is combined with the data masking module it uses the following flow:

1. **tool_calls response:** PII in the template is pseudonymized before reaching the LLM. Tool call arguments returned by the LLM contain pseudonymized placeholders (e.g. `MASKED_PERSON_1`). The output unmasking step restores the original PII values in the `tool_calls` arguments before returning them to the client.
2. **stop response:** When you send the unmasked assistant message back (with original PII in `tool_calls[].function.arguments`), input masking automatically re-pseudonymizes the arguments before they reach the LLM. The final stop response is then unmasked as usual.

> **Note**
>
> Unmasking of tool call arguments only applies when using `pseudonymization`. With `anonymization`, the original values are irreversibly removed and tool call arguments will contain masked placeholders. Therefore in case of `pseudonymization`, you can safely use them to call your external functions with the original data.

Content Filtering

When tool calling is combined with the content filtering module it uses the following flow:

1. **Input filtering:** Filters text content from user, system, and tool messages. Assistant message `content` (typically empty in tool_call responses) is also filtered. However, `tool_calls[].function.arguments` are **not** filtered.
2. **Output filtering:** Filters the LLM response `content`. Tool call arguments in the response pass through unfiltered.

Translation

When tool calling is combined with the translation module it uses the following flow:

- **Input translation:** Translates text content from user, system, and tool messages. Tool call arguments in assistant messages are **not** translated, since they typically contain structured data (e.g. JSON) that should not be altered.
- **Output translation:** Translates the LLM response `content`. For Phase 1 tool_call responses the content is typically empty, so output translation is effectively a no-op. It becomes meaningful in Phase 2 when the LLM produces a plain-text stop response.

1.4.3.2.2.6 Defining Tools through Model Parameters

In addition to defining tools in the template's `tools` array, you can define tools in `model.params.tools`. This approach is useful when the tool list is assembled dynamically at runtime instead of being stored in a template.

The tools defined in `model.params.tools` are included as part of the model configuration and are available during request processing.

```
{
  "model": {
    "name": "gpt-4o",
    "params": {
      "tools": [
        {
          "type": "function",
          "function": {
            "name": "get_inventory_quantity",
            "description": "Check available quantity for a product ID.",
```

```
            "parameters": {
              "type": "object",
              "properties": {
                "product_id": {
                  "type": "integer"
                }
              },
              "required": [
                "product_id"
              ]
            }
          }
        }
      ],
      "tool_choice": "auto"
    }
  }
}
```

Tool Precedence

When `model.params.tools` is specified, it takes precedence over any tools defined in the template.

Orchestration does not merge tools from the template and `model.params.tools`. Template-defined tools are used only when the `model.params` object does not contain a `tools` key.

> **Note**
>
> When you use `model.params.tools`, specify additional tool-related parameters, such as `tool_choice` and `parallel_tool_calls`, in the `model.params` object.

1.4.3.2.2.7 Example of a Partial Orchestration Config

Orchestration configs that lack mandatory fields (such as model or prompt details) can still be created, but the mandatory fields must be provided at time of consumption in Orchestration. In the following example, the prompt is missing:

```
curl -X POST "$AI_API_URL/v2/registry/v2/orchestrationConfigs" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $TOKEN" \
--data '{
    "name": "partial-orchestration-config",
    "version": "0.0.1",
    "scenario": "customer-support",
    "spec": {
      "modules": {
        "prompt_templating": {
          "model": {
            "name": "gpt-4o"
          }
        }
      }
    }
  }'
```

The following consumption example provides the mandatory prompt and an optional tool definition:

```
curl --request POST \
  --url $ORCHESTRATION_DEPLOYMENT_URL/v2/completion \
  --header "AI-Resource-Group: $RESOURCE_GROUP" \
  --header 'authorization: Bearer {{TOKEN}}' \
  --header 'content-type: application/json' \
  --data '{
  "config_ref": {
    "id": "b5ea5b32-0151-4608-98f6-d260dba6700d"
  },
  "config": {
    "modules": {
      "prompt_templating": {
        "prompt": {
          "template": [
            {
              "role": "user",
              "content": "What is the weather in Potsdam and in Toulouse in
Celsius?"
            }
          ],
          "defaults": {},
          "tools": [
            {
              "type": "function",
              "function": {
                "description": "Get weather for location",
                "name": "getCurrentWeather",
                "parameters": {
                  "$schema": "https://json-schema.org/draft/2020-12/schema",
                  "additionalProperties": false,
                  "type": "object",
                  "properties": {
                    "arg0": {
                      "type": "object",
                      "properties": {
                        "location": {
                          "type": "string"
                        },
                        "unit": {
                          "type": "string",
                          "enum": [
                            "C",
                            "F"
                          ]
                        }
                      },
                      "required": [
                        "location",
                        "unit"
                      ]
                    }
                  },
                  "required": [
                    "arg0"
                  ]
                },
                "strict": false
              }
            }
          ]
        }
      }
    }
  }
}'
```

1.4.3.2.3 Translation

The translation module is optional. It allows you to translate LLM text prompts into a chosen target language.

Input Translation

The input translation module helps improve answer quality when the configured model performs better with input in a specific language, such as English.

Output Translation

The output translation module is optional. It translates LLM output into a chosen target language.

Configuration

To use translation, configure it as part of your orchestration workflow.

The module supports SAP's Document Translation service, which is part of the SAP Translation Hub.

> ⚠ Caution
>
> Input translation is performed before any configured input masking. Any content added to the prompt as part of prompt templating is sent to the translation service unmodified. Output translation is performed after any configured output unmasking: unmasked output is sent to the translation service.

Related Information

Enhance Model Consumption with Input Translation [page 543]
Enhance Model Consumption with Output Translation [page 548]

1.4.3.2.3.1 Enhance Model Consumption with Input Translation

To configure translation, add a `translation` section to the `modules` of your `config`.

Configuration Example

Sample Code

```
    "config": {
        "modules": {
            ...
            "translation": {
                "input": {
                    "type": "sap_document_translation",
                    "config": {
                        "source_language": "de-DE",
                        "target_language": "en-US"
                    }
                }
            }
            ...
        }
    }
    ...
}
...
```

Configuration Parameters

The `translation` object must specify the keys::

- `type`: the translation service being called.
  The following values are supported:
  - `sap_document_translation`
- `config`: parameters sent to the service to configure translation.
  The following parameters are supported:
  - `source_language` (optional): the language of the input text. If no source language is specified, the SAP Document Translation Service automatically detects the source language.
  - `target_language` (mandatory): the language to translate the input to.
  - `apply_to` (optional): specify which parts of the input to translate for granular control. For more information , see Selective Translation with apply_to [page 545].

Note

For texts containing multiple languages, the service might be unable to detect the source language accurately.

Supported Languages

For more information on supported languages and translation pairs, see Supported Languages of the document translation service in SAP Translation Hub.

Selective Translation with `apply_to`

The `apply_to` parameter enables precise control over which content gets translated, allowing you to translate only specific parts of your input while leaving others unchanged.

Configuration Properties

Each entry in the `apply_to` array supports the following:

- `category` (required): the content type to select for translation
  Supported values:
  - `placeholders`: translate specific placeholder values from `placeholder_values` and `grounding.placeholders.output`
  - `template_roles`: translate template messages with specific message roles
- `items` (required): an array of specific items within the category to translate
  - For `category: placeholders`: array of placeholder keys
    - Examples: `["user_input", "grounding_input", "grounding_output", "custom_parameter"]`
  - For `category: template_roles`: Array of message role names from your template.
    - Supported values: `["user", "assistant", "system", "developer", "tool"]`
    - These correspond to the `role` field in your message objects
- `source_language` (optional): The source language for this entry. If not provided, the service automatically detects the source language.

> → Tip
>
> When using automatic source language detection, avoid grouping items with different languages in the same `apply_to` entry, as this may lead to incorrect translation results. Use separate entries for items in different languages.

> {·} Sample Code

```
{
    "config": {
        "modules": {
            "translation": {
                "input": {
                    "type": "sap_document_translation",
                    "config": {
                        "target_language": "en-US",
                        "apply_to": [
                            {
                                "category": "placeholders",
                                "items": [
                                    "user_input"
                                ],
                                "source_language": "de-DE"
                            },
                            {
                                "category": "placeholders",
                                "items": [
                                    "grounding_output"
                                ]
                            },
                            {
                                "category": "template_roles",
```

```
                                "items": [
                                    "user",
                                    "assistant"
                                ]
                            }
                        ]
                    }
                }
            }
        }
    }
}
```

Chat History Control

By default, chat history messages are translated. To avoid repeating the translation of chat history messages that have already been translated, set `translation.input.config.translate_messages_history` to `false`.

Examples

Example 1: Customer Support Request Translation

The following example shows a summarizing support assistant and uses the translation module to translate a support request message from German to English before sending it to the LLM for summarization.

Sample Code

```
{
    "config": {
        "modules": {
            "prompt_templating": {
                "messages_history": [
                    {
                        "role": "system",
                        "content": "You are a helpful support assistant. Your
task is to summarize a given support request for the human support team.\nYou
proceed as follows: Summarize the issue for the human support team. Provide
your answer in the following format:\n - Sentiment: [your sentiment analysis]
\n - Key Theme: [theme of the support issue]\n - Contact: [any contact
information available in the issue]"
                    },
                    {
                        "role": "user",
                        "content": "Support Request: '''{{?support-
request}}'''"
                    }
                ]
            },
            "model": {
                "name": "gpt-4o-mini"
            },
            "translation": {
                "input": {
                    "type": "sap_document_translation",
```

```
                    "config": {
                        "source_language": "de-DE",
                        "target_language": "en-US"
                    }
                }
            }
        }
    },
    "placeholder_values": {
        "support-request": "Subject: Bestellung #1234567890 Verspätet - John
Johnson Nachricht: Hallo, ich schreibe ihnen um mich nach dem Status meiner
Bestellung mit der Bestellnr. +1234567890 zu erkundigen. Die Lieferung war
eigentlich für gestern geplant, ist bisher jedoch nicht erfolgt. Mein Name
ist John Johnson und meine Lieferadresse lautet 125 Cole Meadows Drive Palo
Alto, California 94301. Bitte lassen Sie mich per Telefon unter der Nummer +1
505802 2172 wissen, wann ich mit meiner Lieferung rechnen kann. Danke!"
    }
}
```

Example 2: Study Guide with Selective Placeholder Translation

A study guide generator that translates specific input parameters while keeping others unchanged:

Sample Code

```
 {
    "placeholder_values": {
        "exam_type": "Abitur",
        "topic": "Deutsche Literatur",
        "num_questions": "5"
    },
    "config": {
        "modules": {
            "prompt_templating": {
                "prompt": {
                    "template": [
                        {
                            "role": "system",
                            "content": "You are an expert study coach
creating clear, concise exam notes and practice questions."
                        },
                        {
                            "role": "user",
                            "content": "Generate a study guide for the
{{?exam_type}} exam on {{?topic}}.\n\nInclude {{?num_questions}} practice
questions."
                        }
                    ]
                },
                "model": {
                    "name": "gpt-4o"
                }
            },
            "translation": {
                "input": {
                    "type": "sap_document_translation",
                    "config": {
                        "apply_to": [
                            {
                                "category": "placeholders",
                                "items": [
                                    "exam_type",
                                    "topic"
                                ],
                                "source_language": "de-DE"
```

```
                            }
                        ],
                        "target_language": "en-US"
                    }
                }
            }
        }
    }
}
```

1.4.3.2.3.2 Enhance Model Consumption with Output Translation

Output translation translates LLM responses into your preferred language. This ensures the LLM can operate at peak performance in its strongest language while delivering responses in the format most useful to the user.

To configure output translation, add a `translation` section to the `modules` of your `config`.

Configuration Example

Sample Code

```
{
   "config": {
       "modules": {
           ...
           "translation": {
               "output": {
                   "type": "sap_document_translation",
                   "config": {
                       "source_language": "en-US",
                       "target_language": "de-DE"
                   }
               }
           }
           ...
       }
   }
   ...
}
...
```

Configuration Parameters

The `translation` must object must specify the keys:

- `type`: the translation service being called

The following values are supported:

- `sap_document_translation`

- `config`: the parameters that are sent to the service to configure translation
- `target_language` (mandatory): the output language
  The following values are supported:
  - A language code string (for example, "de-DE") for fixed target language
  - A reference object to match the language of your input query. For more information, see Automatic Target Language Inference [page 549].

Supported Languages

For more information on supported languages and translation pairs, see Supported Languages of the document translation service in SAP Translation Hub.

Automatic Target Language Inference

Output translation can automatically respond in the user's original language by referencing the detected source language from input translation. This is particularly useful if the user's language is not known in advance.

Instead of specifying a fixed target language, reference an `apply_to` item from your input translation. For more information, see Selective Translation with apply_to [page 545].

Sample Code

```
{
    "config": {
        "modules": {
            "translation": {
                "input": {
                    "type": "sap_document_translation",
                    "config": {
                        "target_language": "en-US",
                        "apply_to": [
                            {
                                "category": "template_roles",
                                "items": [
                                    "user"
                                ]
                            }
                        ]
                    }
                },
                "output": {
                    "type": "sap_document_translation",
                    "config": {
                        "source_language": "en-US",
                        "target_language": {
                            "category": "template_roles",
                            "items": [
                                "user"
                            ]
```

```
                        }
                    }
                }
            }
        }
    }
}
```

⚠ Restriction

The reference must exactly match an existing `apply_to` entry from your input translation configuration. In the example above, the `target_language` object in the output translation matches the first element in the input translation's `apply_to` array.

Examples

Example 1: Basic Output Translation

The following example shows a summarizing support assistant and uses the translation module to translate a support request summary into German for a German-speaking support team.

{·} Sample Code

```
{
    "config": {
        "modules": {
            "prompt_templating": {
                "messages_history": [
                    {
                        "role": "system",
                        "content": "You are a helpful support assistant. Your
task is to summarize a given support request for the human support team in
Germany."
                    },
                    {
                        "role": "user",
                        "content": "Support Request: '''{{?support-
request}}'''"
                    }
                ]
            },
            "model": {
                "name": "gpt-4o-mini"
            },
            "translation": {
                "output": {
                    "type": "sap_document_translation",
                    "config": {
                        "source_language": "en-US",
                        "target_language": "de-DE"
                    }
                }
            }
        }
    },
    "placeholder_values": {
        "support-request": "Subject: Order #1234567890 Delayed - John Johnson
Message: Hi I'm writing to inquire about the status of my order +1234567890.
It was supposed to arrive yesterday but hasn't shown up. My Name is John
```

```
Johnson, and the delivery address is 125 Cole Meadows Drive Palo Alto,
California 94301. Please advice on the expected delivery date via +1 505802
2172. Thanks!"
    }
}
```

Example 2: Automatic Target Language Matching

A travel assistant that automatically responds in the same language as the input:

Sample Code

```
{
    "config": {
        "modules": {
            "prompt_templating": {
                "prompt": {
                    "template": [
                        {
                            "role": "system",
                            "content": "You are a helpful travel assistant
who gives clear and friendly planning advice."
                        },
                        {
                            "role": "user",
                            "content": "Ich möchte nächsten Monat nach Paris
reisen. Kannst du mir bei der Planung helfen?"
                        }
                    ]
                },
                "model": {
                    "name": "gpt-4o"
                }
            },
            "translation": {
                "input": {
                    "type": "sap_document_translation",
                    "config": {
                        "apply_to": [
                            {
                                "category": "template_roles",
                                "items": [
                                    "user"
                                ]
                            }
                        ],
                        "target_language": "en-US"
                    }
                },
                "output": {
                    "type": "sap_document_translation",
                    "config": {
                        "source_language": "en-US",
                        "target_language": {
                            "category": "template_roles",
                            "items": [
                                "user"
                            ]
                        }
                    }
                }
            }
        }
    }
}
```

1.4.3.2.4 Data Masking

The data masking module is optional and serves to anonymize or pseudonymize personally identifiable information from the input for selected entities.

There are two ways in which you can mask data:

- Anonymization replaces personally identifiable information in chosen categories with a `MASKED_ENTITY` placeholder. Anonymized data can't be unmasked because the original data isn't retained.
- Pseudonymization substitutes personally identifiable information in selected categories with a `MASKED_ENTITY_ID` placeholder. Pseudonymized data can be unmasked in the response.

The module currently supports the following anonymization services:

- SAP Data Privacy Integration - Anonymization

SAP Data Privacy Integration - Anonymization

SAP Data Privacy Integration - Anonymization service recognizes the following entity categories:

Entity	Description	Scope
profile-person	Person Name	English
profile-org	Organization	SAP customers + Fortune 1000 companies
profile-university	University	Organization identified as a public University
profile-location	Location	United States
profile-phone	Phone Number	International phone numbers in the following format: +country/region code, any prefix and phone number
profile-address	U.S. Physical Addresses	United States
profile-email	E-mail addresses	E-mail addresses
profile-sapids-internal	SAP Staff User ID Numbers	User ID: Sequence of characters that starts with either C, I, or D followed by 6-8 digits.
profile-sapids-public	Public SAP Accounts	S-user ID: Sequence of characters that starts with S and followed by 6-11 digits. P-user ID: Sequence of characters that starts with P followed by 10 digits.
profile-url	URL	Only anonymize URLs that can be accessed by the user
profile-username-password	Username and Passwords	Detected if words similar to user, user name, pass, passwords, and so on.
profile-nationalid	National ID Number	20+ countries/regions (for example, E.U., South America)
profile-iban	International Bank Account Number (IBAN)	70+ countries/regions (E.U., Middle East, South America)

Entity	Description	Scope
profile-ssn	Social Security Number (SSN)/Social Insurance Number (SIN)	United States and Canada
profile-credit-card-number	Credit Card Number	Global
profile-passport	Passport Number	30+ countries/regions (for example, E.U., Asia Pacific, North America)
profile-driverlicense	Driver's License	30+ countries/regions (for example, E.U., Asia Pacific)
profile-nationality	Nationality	190+ country/region names, official country/region names, and country/region codes
profile-religious-group	Religious Group	200+ religious groups
profile-political-group	Political Parties or Groups	100+ political parties or groups
profile-pronouns-gender	Gender pronoun	Global
profile-gender	Gender	Global
profile-sexual-orientation	Sexual Orientation	Global
profile-trade-union	Trade Union	Global
profile-sensitive-data	Nationality, Religious_Group, Gender, Political_Group, pronoun gender, ethnicity, sexual orientation, Trade union	Mixed
profile-ethnicity	Ethnicity	Global

To mask entities based on regular expressions, use custom entities. For more information, see Configuring Replacement Methods [page 369].

> ⚠ Caution
>
> The masking service can obscure personally identifiable information in prompts. However, since it relies on automated detection mechanisms, it can't guarantee that all such information is identified and masked.
>
> Anonymization replaces personally identifiable information in an irreversible way. As a result, context can be lost, which can limit the model's ability to process the input. For instance, if tasked with writing a story about Michael and Donna, anonymization of profile-person results in a story about `MASKED_PERSON` and `MASKED_PERSON`, making it impossible to distinguish between the two.

Related Information

Access the Data Masking using the SAP AI SDK for Java
Access the Data Masking using the SAP AI SDK for JavaScript

1.4.3.2.4.1 Enhancing Model Consumption with Data Masking

Prerequisites

You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].

Process

In the following example, we use the data masking module to anonymize people, organizations, and contact information in the input. . Here, input is masked before the model is called. The data in the model's output cannot be unmasked.

Sample Code

```
curl --request POST $ORCH_DEPLOYMENT_URL/v2/completion \
    --header 'content-type: application/json' \
    --header "Authorization: Bearer $TOKEN" \
    --header "AI-Resource-Group: $RESOURCE_GROUP" \
    --data-raw '{
      "config": {
        "modules": {
          "prompt_templating": {
            "prompt": {
              "template": [
              {
                "role": "user",
                "content": "{{?input}}"
              }
            ]
            },
            "model": {
              "name": "gpt-4o",
              "params": {
                "max_completion_tokens": 300,
                "temperature": 0.1,
                "frequency_penalty": 0,
                "presence_penalty": 0
              }
          }
          },
        "masking": {
          "providers": [
            {
              "type": "sap_data_privacy_integration",
              "method": "anonymization",
              "entities": [
                {"type": "profile-person"},
                {"type": "profile-email"},
                {"type": "profile-phone"}
              ]
            }
          ]
```

```
        }
        }
      },
      "placeholder_values": {
        "input": "Repeat the following: Penny lives with her partner Peter
and her dog Coco. Penny can be reached via pennylovesdogs@example.com and her
phone number is +91-9899999999"
      }
}'
```

> **Note**
>
> The `masking_providers` parameter is deprecated, and has been replaced by `providers`. Users with configurations that include `masking_providers` should update them.

As shown in the response, the data masking module masks the configured entities before sending them to the model. The model then operates on this masked data and can still provide a summary.

> **Caution**
>
> The `intermediate_results` field may contain sensitive information. We recommend not displaying its contents to end users.

Output Code

```
{
  "request_id": "cd6fcf22-1d21-9b82-a3cd-fc896fe41e14",
  "intermediate_results": {
    "templating": [
      {
        "content": "Repeat the following: Penny lives with her partner Peter
and her dog Coco. Penny can be reached via pennylovesdogs@example.com and her
phone number is +91-9899999999",
        "role": "user"
      }
    ],
    "input_masking": {
      "message": "Masking module executed successfully on LLM input.",
      "data": {
        "masked_template": "[{\"content\": \"Repeat the following:
MASKED_PERSON lives with her partner MASKED_PERSON and her dog Coco.
MASKED_PERSON can be reached via MASKED_PERSONMASKED_EMAIL and her phone
number is MASKED_PHONE_NUMBER\", \"role\": \"user\"}]"
      }
    },
    "llm": {
      "id": "chatcmpl-BtrVHn0HeUEnwHCIpjuReoYfgJfD8",
      "object": "chat.completion",
      "created": 1752653407,
      "model": "gpt-4o-2024-08-06",
      "system_fingerprint": "fp_ee1d74bde0",
      "choices": [
        {
          "index": 0,
          "message": {
            "role": "assistant",
            "content": "I'm sorry, I can't assist with that request."
          },
          "finish_reason": "stop"
        }
      ],
      "usage": {
        "completion_tokens": 11,
```

```
        "prompt_tokens": 48,
        "total_tokens": 59
      }
    }
  },
  "final_result": {
    "id": "chatcmpl-BtrVHn0HeUEnwHCIpjuReoYfgJfD8",
    "object": "chat.completion",
    "created": 1752653407,
    "model": "gpt-4o-2024-08-06",
    "system_fingerprint": "fp_ee1d74bde0",
    "choices": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": "I'm sorry, I can't assist with that request."
        },
        "finish_reason": "stop"
      }
    ],
    "usage": {
      "completion_tokens": 11,
      "prompt_tokens": 48,
      "total_tokens": 59
    }
  }
}
```

Optionally, you can include an `allowlist` in your request. Items on your `allowlist` are excempt from data masking.

In the following example, the allow list contains the entries “Penny” and “Peter”, therefore these values are not masked.

Sample Code

```
curl --request POST $ORCH_DEPLOYMENT_URL/v2/completion \
    --header 'content-type: application/json' \
    --header "Authorization: Bearer $TOKEN" \
    --header "AI-Resource-Group: $RESOURCE_GROUP" \
    --data-raw '{
      "config": {
        "modules": {
          "prompt_templating": {
            "prompt": {
              "template": [
              {
                "role": "user",
                "content": "{{?input}}"
              }
            ]
            },
            "model": {
              "name": "gpt-4o",
              "params": {
                "max_completion_tokens": 300,
                "temperature": 0.1,
                "frequency_penalty": 0,
                "presence_penalty": 0
              }
          }
          },
        "masking": {
          "providers": [
```

```
            {
              "type": "sap_data_privacy_integration",
              "method": "anonymization",
              "entities": [
                {"type": "profile-person"},
                {"type": "profile-email"},
                {"type": "profile-phone"}
              ],
              "allowlist": ["Penny", "Peter"]
          }
          ]
        }
        }
      },
      "placeholder_values": {
        "input": "Repeat the following: Penny lives with her partner Peter
and her dog Coco. Penny can be reached via pennylovesdogs@example.com and her
phone number is +91-9899999999"
      }
}'
```

When you use `"method": "pseudonymization"` instead of `"method": "anonymization"` in the masking module configuration, the data is pseudonymized before it's sent to the model. Additionally, the system checks the model's response for any data that can be unmasked before sending out the final response.

Output Code

```
{
  "request_id": "9a5f3241-5ff9-9011-bfc9-7cd9d993788d",
  "intermediate_results": {
    "templating": [
      {
        "content": "Repeat the following: Penny lives with her partner Peter
and her dog Coco. Penny can be reached via pennylovesdogs@example.com and her
phone number is +91-9899999999",
        "role": "user"
      }
    ],
    "input_masking": {
      "message": "Masking module executed successfully on LLM input.",
      "data": {
        "masked_template": "[{\"content\": \"Repeat the following:
MASKED_PERSON_2 lives with her partner MASKED_PERSON_3 and her dog Coco.
MASKED_PERSON_2 can be reached via MASKED_PERSON_1MASKED_EMAIL_1 and her
phone number is MASKED_PHONE_NUMBER_1\", \"role\": \"user\"}]"
      }
    },
    "llm": {
      "id": "chatcmpl-BuaBiRJcbI0NSglMoTl5xRtQUeUor",
      "object": "chat.completion",
      "created": 1752825174,
      "model": "gpt-4o-2024-08-06",
      "system_fingerprint": "fp_ee1d74bde0",
      "choices": [
        {
          "index": 0,
          "message": {
            "role": "assistant",
            "content": "MASKED_PERSON_2 lives with her partner
MASKED_PERSON_3 and her dog Coco. MASKED_PERSON_2 can be reached via
MASKED_PERSON_1MASKED_EMAIL_1 and her phone number is MASKED_PHONE_NUMBER_1"
          },
          "finish_reason": "stop"
        }
      ],
```

```
      "usage": {
        "completion_tokens": 50,
        "prompt_tokens": 60,
        "total_tokens": 110
      }
    },
    "output_unmasking": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": "Penny lives with her partner Peter and her dog Coco.
Penny can be reached via pennylovesdogspennylovesdogs@example.com and her
phone number is +91-9899999999"
        },
        "finish_reason": "stop"
      }
    ]
  },
  "final_result": {
    "id": "chatcmpl-BuaBiRJcbI0NSglMoTl5xRtQUeUor",
    "object": "chat.completion",
    "created": 1752825174,
    "model": "gpt-4o-2024-08-06",
    "system_fingerprint": "fp_ee1d74bde0",
    "choices": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": "Penny lives with her partner Peter and her dog Coco.
Penny can be reached via pennylovesdogspennylovesdogs@example.com and her
phone number is +91-9899999999"
        },
        "finish_reason": "stop"
      }
    ],
    "usage": {
      "completion_tokens": 50,
      "prompt_tokens": 60,
      "total_tokens": 110
    }
  }
}
```


When the grounding module is used, the masking module also supports masking of the grounding input before retrieval. By default, this feature is disabled. To enable it the parameter `mask_grounding_input` in the `masking` can be used as follows:

Sample Code


```
{
  "config": {
    "modules": {
      "masking": {
        "providers": [
          {
            "type": "sap_data_privacy_integration",
            "method": "pseudonymization",
            "entities": [
              {"type": "profile-person"},
              {"type": "profile-email"}
            ],
            "mask_grounding_input": {"enabled": true}
          }
```

```
        ]
        ...
      }
      ...
    }
  }
}
```

Configuring Replacement Methods

You can configure the replacement method for each entity in the `masking`. The following methods are supported:

- `fabricated_data`: Replaces the entity with a randomly generated value of the same type. For example, a person's name might be replaced with another name, and a phone number with a different phone number.
- `constant`: Replaces the entity with a specified constant value.
  - In anonymization, the constant value is used directly without any modification (for example, `MASKED_PERSON`).
  - In pseudonymization, the constant is appended with an incrementing number to ensure uniqueness (for example, `MASKED_PERSON_1`, `MASKED_PERSON_2`).

If no replacement strategy is specified, the default `constant` replacement method is used with a default constant value (for example, `MASKED_PERSON` for person entities).

The following example shows code for fabricated data replacement:

```
"entities": [
  {
    "type": "profile-person",
    "replacement_strategy": {
      "method": "fabricated_data"
    }
  }
]
```

The following example shows code for constant replacement with a custom value:

```
"entities": [
  {
    "type": "profile-person",
    "replacement_strategy": {
      "method": "constant",
      "value": "PERSON_REDACTED"
    }
  }
]
```

Using Custom Entities

In addition to the standard entity types, you can define custom entities using regular expressions. This allows you to mask specific patterns in your data that are not covered by the standard entity types. Custom entities require specifying the `constant` replacement strategy.

The following example shows code for constant replacement with a custom entity:

```
"entities": [
```

```
  {
    "regex": "[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\\.[a-zA-Z0-9-.]+",
    "replacement_strategy": {
      "method": "constant",
      "value": "REDACTED_EMAIL"
    }
  }
]
```

In this example, the regular expression `[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\\.[a-zA-Z0-9-.]+` is used to identify email addresses. The identified email addresses will be replaced with `REDACTED_EMAIL_1`, `REDACTED_EMAIL_2`, and so on.

PDF Input

PDF input is supported by the masking module. If file content is included and masking is configured, the parameter `mask_file_input_method` must be set to `anonymization` or `skip`.

```
{
  "config": {
    "modules": {
      "masking": {
        "providers": [
          {
            "type": "sap_data_privacy_integration",
            "method": "pseudonymization",
            "entities": [
              {"type": "profile-person"},
              {"type": "profile-email"}
            ],
            "mask_file_input_method": "anonymization"
          }
        ]
        ...
      }
      ...
    }
  }
}
```

Related Information

Leveraging Orchestration Capabilities to Enhance Responses
Libraries and SDKs

1.4.3.2.5 Content Filtering

The content filtering module lets you filter input and output based on content safety criteria. The module is optional. If you don't configure an input filter in the orchestration settings, the input goes to the model

configuration without filtering. If you don't configure an output filter in the orchestration settings, the response returns without any filtering.

The module supports two services:

- Azure Content Safety
- Llama Guard 3
- Granite Guardian 4

The content filtering module allows you to configure multiple distinct filters for a single request. However, each filter type (such as Azure Content Safety, Llama Guard,) can be configured only once, meaning that you can't set up two filters of the same type but with different settings. However, different filter types can be applied concurrently. The orchestration service waits for all content filters to complete before returning the results.

> **Note**
>
> Do not add explicit content filtering rules to prompts when using the content filtering module. Including examples of restricted content can affect content classification and may cause false positives.

> **Note**
>
> For all Azure OpenAI models, a global content filter is configured on the Azure AI platform. This global filter removes all input and output that is classified as medium (4) or high (6) in any of the categories.

Azure Content Safety

The Azure Content Safety service provides the following content filtering capabilities:

- Harm Classification
- Prompt Attack Detection
- Protected Material Detection for Code

Harm Classification

The Azure Content Safety classification service recognizes the following distinct content categories: `Hate`, `Violence`, `Sexual`, and `SelfHarm`. For more information, see Harm categories in Azure AI Content Safety . Text can have more than one label (for example, a text sample can be classified as both `Hate` and `Violence`). The returned content categories include a severity level rating of 0, 2, 4, or 6. The value increases with the severity of the content.

Prompt Attack Detection

The Azure Content Safety service also supports prompt attack detection for input text via `PromptShield` configuration. For more information, see Prompt Shields in Azure AI Content Safety .

A prompt attack is a malicious input that is designed to bypass a model's safety mechanisms or override previous instructions. Prompt attacks can lead to the generation of harmful content or the execution of malicious actions. The Azure Content Safety service detects prompt attacks and returns a Boolean value.

If both prompt attack detection and harm classification are configured, the orchestration service performs the prompt attack detection call first, and then performs the harm classification call. If a prompt attack is

detected, the orchestration service doesn't make the harm classification request and returns the prompt attack detection result only.

> **Note**
>
> Content from messages with the `system` and `developer` roles is excluded from the `prompt_shield` content filter. As a result, if a request contains only `system` or `developer` messages, the `prompt_shield` content filter is skipped because there's no content to evaluate.

Protected Material Detection for Code

The Azure Content Safety service includes an output content filter that identifies code matching existing GitHub repository code.

The filter scans publicly available code, and has a knowledge cutoff date. Only code matching publicly available code added to GitHub after the knowledge cutoff date can be detected.

The filter requires input text longer than 110 characters (minimum 111 characters). This limitation is defined by Azure and cannot be modified. Requests containing 110 characters or fewer return an HTTP `400 Bad Request` response. For streaming requests, configure both `chunk_size` and `overlap` so that the minimum character requirement is met.

For more information, see Streaming [page 581] and Protected Material Detection for Code in Azure AI Content Safety .

> **→ Remember**
>
> When multiple content filtering types are configured together, each type is processed as a separate request. Costs are incurred for each filtering type applied.

> **Note**
>
> To ensure continued service availability, the orchestration service supports regional fallbacks for the Azure Content Safety service. If a request fails due to service unavailability, the orchestration service automatically retries the request using a fallback endpoint within the same region.

Llama Guard 3

Llama Guard 3 recognizes the following content categories:

- `violent_crimes`
- `non_violent_crimes`
- `sex_crimes`
- `child_exploitation`
- `defamation`
- `specialized_advice`
- `privacy`
- `intellectual_property`
- `indiscriminate_weapons`

- `hate`
- `self_harm`
- `sexual_content`
- `elections`
- `code_interpreter_abuse`

For more information, see Hazard Taxonomy and Policy in Llama Guard 3 8B . Texts can have multiple labels. The returned content categories include a boolean value that shows whether the text contains content triggering the filter for each respective category.

> **Note**
>
> Llama Guard 3 8B can report violations for categories that haven't been requested by the user
>
> The orchestration service doesn't consider these violations when processing Llama Guard responses, and only filters violations in categories that have been requested by the user.

Granite Guardian 4

Granite Guardian 4 recognizes the following content categories:

- `harm`
- `social_bias`
- `jailbreak`
- `violence`
- `profanity`
- `sexual_content`
- `unethical_behavior`

Granite Guardian 4 also includes a reasoning mode that provides a rationale for the classification decision. For more information, see Granite Guardian 4 Prompting Guide .

The returned content categories include a Boolean value that shows whether or not the text contains content triggering the filter for each respective category. If reasoning mode is enabled, the response also includes reasoning content.

Related Information

Enhancing Model Consumption with Input Filtering [page 564]
Enhancing Model Consumption with Output Filtering [page 568]

1.4.3.2.5.1 Enhancing Model Consumption with Input Filtering

Prerequisites

You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].

Process

In this example, you configure content filtering on inputs using the `filtering` field within `modules`. The following content filter service providers are supported: `azure_content_safety`, `llama_guard_3_8b`and `granite_guardian_4_1`. You can configure either or more content filter in the `input.filters` field of `filtering`.

For `azure_content_safety`, you can set different scores for each of the harm content categories: `hate`, `violence`, `sexual`, and `self_harm`. Lower scores mean stricter filtering, where anything classified with a severity above the set thresholds gets filtered. Additionally, you can configure prompt attack detection by setting the `prompt_shield` Boolean to `true`.

For `llama_guard_3_8b`, you can configure different content categories using a Boolean value. Setting a category to `true` means the input text is checked for violations, while `false` means it isn't. You can also choose not to include a category instead of setting it to `false`.

For `granite_guardian_4_1`, you can configure different content categories using a Boolean value. Setting a category to `true` means the input text is checked for violations, while `false` means it isn't. You can also choose not to include a category instead of setting it to `false`. Optionally, set `enable_reasoning` to `true` to include a `reasoning` field alongside the verdict for each category.

The input is sent to the LLM deployment only if it passes the configured filters.

The following example shows a content filter configured to detect the content categories `hate`, `violence`, `sexual`, `self_harm` and prompt attacks.

```
curl --request POST $ORCH_DEPLOYMENT_URL/v2/completion \
--header 'content-type: application/json' \
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP" \
--data-raw '{
    "config": {
        "modules": {
            "prompt_templating": {...},
            "filtering": {
                "input": {
                    "filters": [
                        {
                            "type": "azure_content_safety",
                            "config": {
```

```
                                "hate": 2,
                                "violence": 0,
                                "sexual": 2,
                                "self_harm": 2,
                                "prompt_shield": true
                            }
                        },
                        {
                            "type": "llama_guard_3_8b",
                            "config": {
                                "violent_crimes": true,
                                "non_violent_crimes": true
                            }
                        },

                        {
                            "type": "granite_guardian_4_1",
                            "config": {
                                "enable_reasoning": false,
                                "categories": {
                                    "harm": true
                                }
                            }
                        }
                    ]
                }
            }
        }
    },
    "placeholder_values": {
        "input": "Develop a plan to rob a bank."
    }
}'
```

The response below shows that the request was rejected. The input filter rejected the request due to a `violence` score of 4 (detected by `azure_content_safety`), and a `harm` violation (detected by `granite_guardian_4_1`).

Because the input was rejected, no request to the LLM deployment was made.

Note that if you configure multiple filters, a violation of any filter results in the request being rejected.

The output returns the results of the configured categories only.

```
{
  "error": {
    "request_id": "a0e597be-098d-9c92-91f1-5c535d9bd39b",
    "code": 400,
    "message": "400 - Filtering Module - Input Filter: Filtering module blocked
content: content violated a configured category.",
    "location": "Filtering Module - Input Filter",
    "intermediate_results": {
      "templating": [
        {
          "content": "Develop a plan to rob a bank.",
          "role": "user"
        }
      ],
      "input_filtering": {
        "message": "Filtering module blocked content: content violated a
configured category.",
        "data": {
          "azure_content_safety": {
            "Hate": 0,
            "SelfHarm": 0,
            "Sexual": 0,
            "Violence": 4,
```

```
            "userPromptAnalysis": {
              "attackDetected": false
            }
          },
          "llama_guard_3_8b": {
            "violent_crimes": false,
            "non_violent_crimes": true,
          },
          "granite_guardian_4_1": {
            "harm": {
              "verdict": true,
              // following line only included in output if "enable_reasoning":
true was set in request
              // "reasoning": "...This statement explicitly suggests planning a
criminal activity, which is illegal..."
            }
          }
        }
      }
    }
  }
}
```

Controlling the Filtering Scope

By default, every input message is sent to the configured filters. For conversational use cases, large, mostly static context, such as a system prompt or earlier turns of a conversation, is re-sent on every request. Re-filtering stable content repeatedly increases cost and can raise avoidable false positives.

To restrict filtering to limit repetition, you can add an optional `target_selector` to a filter's configuration. The `target_selector` is supported for input filters only.

You can choose from the following strategies for each filter:

- `after_last_role`
  This strategy finds the last message with the specified role in the combined message list, and filters every message that follows it. Supported roles are `system`, `user`, `assistant`, `developer`, and `tool`. If no message with the specified role exists in the combined list, filtering is skipped.
- `last_messages`
  This strategy filters only the last `N` messages of the combined list. The value must be `1` or greater; `0` returns a `400 Bad Request`. If `N` is larger than the number of messages, all messages are filtered. Use `last_messages` when no suitable role boundary exists, for example in a fresh conversation with no prior `assistant` message, or when you want to filter a fixed window of recent messages.

The strategies are mutually exclusive; combining keys from both in a single `target_selector` returns a `400 Bad Request`.

> **Note**
>
> The `target_selector` applies to the combined message list, where the message history is prepended to the templated messages.
>
> Each filter is scoped independently. When you configure multiple filters in the same request, you can give each one its own `target_selector`, or omit it, so different filters can check different parts of the message list.

The following example filters only the messages that follow the last `assistant` message, so the system prompt and earlier conversation turns are not re-filtered.

```
curl --request POST $ORCH_DEPLOYMENT_URL/v2/completion \
--header 'content-type: application/json' \
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP" \
--data-raw '{
    "config": {
        "modules": {
            "prompt_templating": {...},
            "filtering": {
                "input": {
                    "filters": [
                        {
                            "type": "azure_content_safety",
                            "config": {
                                "hate": 2,
                                "violence": 2,
                                "sexual": 2,
                                "self_harm": 2,
                                "prompt_shield": true
                            },
                            "target_selector": {
                                "after_last_role": "assistant"
                            }
                        }
                    ]
                }
            }
        }
    },
    "placeholder_values": {...}
}'
```

For example, given the following combined message list at request time, the service identifies the last `assistant` message at index 4 and filters only the messages at indices 5 and 6:

```
index 0: { role: "system",    content: "You are a helpful assistant." }
index 1: { role: "user",      content: "What is the capital of France?" }
index 2: { role: "assistant", content: "The capital of France is Paris." }
index 3: { role: "user",      content: "And Germany?" }
index 4: { role: "assistant", content: "The capital of Germany is Berlin." }
(last assistant message)
index 5: { role: "user",      content: "What about Spain?" }
(filtered)
index 6: { role: "tool",      content: "retrieved doc: ..." }
(filtered)
```

To filter a fixed number of the most recent messages instead, use `last_messages` in place of `after_last_role`:

```
"target_selector": {
    "last_messages": 2
}
```

Related Information

Leveraging Orchestration Capabilities to Enhance Responses

Libraries and SDKs

Access the Filtering using the SAP AI SDK for Java

Access the Filtering using the SAP AI SDK for JavaScript

1.4.3.2.5.2 Enhancing Model Consumption with Output Filtering

Prerequisites

You have a running an orchestration deployment and you have retrieved your orchestration deployment URL. For more information, see Get An Orchestration Deployment URL [page 7] and Create a Deployment for Orchestration [page 703].

Process

The following example shows an output content filter for multiple harm categories, and for `protected_material_code`. The output from the LLM is filtered by both `azure_content_safety`, `llama_guard_3_8b`, and `granite_guardian_4_1` before the response is returned.

```
curl --request POST $ORCH_DEPLOYMENT_URL/v2/completion \
--header 'content-type: application/json' \
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP" \
--data-raw '{
    "config": {
        "modules": {
            "prompt_templating": {
                "prompt": {
                    "template": [
                        {
                            "role": "user",
                            "content": "<A prompt that creates output that might
be filtered by output filtering>"
                        }
                    ]
                },
                "model": {...}
            }
        },
        "filtering": {
            "output": {
                "filters": [
                    {
                        "type": "azure_content_safety",
                        "config": {
                            "hate": 0,
                            "self_harm": 0,
                            "sexual": 0,
                            "violence": 0,
```

```
                            "protected_material_code": true
                        }
                    },
                    {
                        "type": "granite_guardian_4_1",
                        "config": {
                            "categories": {
                                "harm": true
                            }
                        }
                    }
                ]
            }
        }
    }
}'
```

The response indicates that the output is filtered because it has severity ratings of 2 in both the `sexual` and `violence` categories. However, it doesn't violate the `granite_guardian_4_1` harm category. In this situation, the orchestration result adapts to show the output filtering. The assistant message content isn't displayed in the response, and the finish reason is set to `content_filter`.

```
{
  "request_id": "4435c6a6-f08e-9561-9a58-1103ac581550",
  "intermediate_results": {
    "templating": [
      {
        "content": "<A prompt that creates output that might be filtered by
output filtering>",
        "role": "user"
      }
    ],
    "output_filtering": {
      "message": "Choice 0: Filtering module blocked content: content violated a
configured category.",
      "data": {
        "choices": [
          {
            "index": 0,
            "azure_content_safety": {
              "Hate": 0,
              "SelfHarm": 0,
              "Sexual": 2,
              "Violence": 0
          },
            "granite_guardian_4_1": {
              "social_bias": {
                "verdict": false
              }
            }
          }
        ]
      }
    },
    "llm": {
      "id": "chatcmpl-Bua9DAmEQjcNASrZL4oWMOtPVlGPm",
      "object": "chat.completion",
      "created": 1752825019,
      "model": "gpt-4o-mini-2024-07-18",
      "system_fingerprint": "fp_efad92c60b",
      "choices": [
        {
          "index": 0,
          "message": {
            "role": "assistant",
            "content": "<A prompt that creates output that might be filtered by
output filtering>"
```

```
          },
          "finish_reason": "stop"
        }
      ],
      "usage": {
        "completion_tokens": 100,
        "prompt_tokens": 100,
        "total_tokens": 200
      }
    }
  },
  "final_result": {
    "id": "chatcmpl-Bua9DAmEQjcNASrZL4oWMOtPVlGPm",
    "object": "chat.completion",
    "created": 1752825019,
    "model": "gpt-4o-mini-2024-07-18",
    "system_fingerprint": "fp_efad92c60b",
    "choices": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": ""
        },
        "finish_reason": "content_filter"
      }
    ],
    "usage": {
      "completion_tokens": 100,
      "prompt_tokens": 100,
      "total_tokens": 200
    }
  }
}
```


> ⚠ Caution
>
> The contents of the returned `module_results` field may include unchecked user or model content. We recommend that you do not display this content to end users.

Related Information

Leveraging Orchestration Capabilities to Enhance Responses
Libraries and SDKs
Access the Filtering using the SAP AI SDK for Java
Access the Filtering using the SAP AI SDK for JavaScript

1.4.3.3 Harmonized API

The Harmonized API lets you use different foundation models without the need to change the client code. It does so by taking the OpenAI API as the standard and mapping other model APIs to it. This mapping includes standardizing message formats, model parameters, and response formats. The Harmonized API is integrated into the templating module, the model configuration, and the orchestration response.

Prompts

In the OpenAI format, a prompt and its response can contain a list of messages with a role and content. The role can be `system`, `user`, or `assistant` and the content is the text of the message. The response can contain a list of choices with `index`, `messages`, and a `finish_reason`.

Example

When you use a Gemini model with orchestration, the OpenAI message format is translated internally to the Vertex AI message format expected by the Gemini model. Consider the following message list in OpenAI format:

```
[
  {
    "role": "system",
    "content": "You are a friendly assistant."
  },
  {
    "role": "user",
    "content": "Hello"
  }
  {
    "role": "assistant",
    "content": "Hi"
  }
]
```

To use this message list with a Gemini model, it's translated internally to the following:

```
{
  "systemInstruction": {
    "role": "system",
    "parts": [
      {
        "text": "You are a friendly assistant."
      }
    ]
  },
  "contents": [
    {
      "role": "user",
      "parts": [
        {
          "text": "Hello"
        }
      ]
    },
    {
      "role": "assistant",
      "parts": [
        {
          "text": "Hi"
        }
      ]
    }
  ]
}
```

Parameters

Parameters: V1

Orchestration supports the model parameters offered by OpenAI, including `max_tokens`, `temperature`, `frequency_penalty`, `presence_penalty`, `n`, and `top_p`. Where possible, these parameters are used with any model and mapped to the corresponding parameter of the foundation model. For example, OpenAI's `max_tokens` is mapped to Vertex AI's `maxOutputTokens`. Additionally, any model-specific parameters can be sent. However, model-specific parameters must be in snake_case format for consistency. For example, you can use Vertex AI's `topK` parameter as `top_k`, which is normally not available in OpenAI's API.

Parameters: V2

Orchestration supports the model parameters offered by OpenAI, including `max_completion_tokens`, `temperature`, `frequency_penalty`, `presence_penalty`, `n`, and `top_p`. Where possible, these parameters are used with any model and mapped to the corresponding parameter of the foundation model. For example, OpenAI's `max_completion_tokens` is mapped to Vertex AI's `maxOutputTokens`. Additionally, any model-specific parameters can be sent. However, model-specific parameters must be in snake_case format for consistency. For example, you can use Vertex AI's `topK` parameter as `top_k`, which is normally not available in OpenAI's API.

Responses from Vertex AI are converted back into OpenAI's format, ensuring the client receives the response in a consistent format. For example, consider the following response in Vertex AI format:


```
{
  "candidates": [
    {
      "content": {
        "parts": [
          {
            "text": "Hello there! How can I assist you today?"
          }
        ],
        "role": "model"
      },
      "finishReason": "STOP",
      "safetyRatings": [
        {
          "category": "HARM_CATEGORY_HATE_SPEECH",
          "probability": "NEGLIGIBLE",
          "probabilityScore": 0.031439852,
          "severity": "HARM_SEVERITY_NEGLIGIBLE",
          "severityScore": 0.04509957
        },
        {
          "category": "HARM_CATEGORY_DANGEROUS_CONTENT",
          "probability": "NEGLIGIBLE",
          "probabilityScore": 0.089933015,
          "severity": "HARM_SEVERITY_NEGLIGIBLE",
          "severityScore": 0.025957357
        },
        {
          "category": "HARM_CATEGORY_HARASSMENT",
          "probability": "NEGLIGIBLE",
          "probabilityScore": 0.053799648,
          "severity": "HARM_SEVERITY_NEGLIGIBLE",
          "severityScore": 0.01711089
        },
        {
          "category": "HARM_CATEGORY_SEXUALLY_EXPLICIT",
          "probability": "NEGLIGIBLE",
```

```
          "probabilityScore": 0.19117664,
          "severity": "HARM_SEVERITY_NEGLIGIBLE",
          "severityScore": 0.046378203
        }
      ]
    }
  ],
  "usageMetadata": {
    "candidatesTokenCount": 10,
    "promptTokenCount": 1,
    "totalTokenCount": 11
  }
}
```

This response is translated internally back into the OpenAI format, returning the following:

```
{
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Hello there! How can I assist you today?"
      },
     "finish_reason": "stop"
    }
  ],
  "usage": {
    "completion_tokens": 10,
    "prompt_tokens": 1,
    "total_tokens": 11
  }
}
```

For information about provider-specific request and response fields, see Provider-Specific Fields [page 577].

OpenAI Responses API

Models Routed Through the OpenAI Responses API

For `gpt-5.3-codex` and all models from `gpt-6` onwards, the orchestration routes requests through the OpenAI Responses API internally. The API uses the same `/v2/completion` endpoint with the chat completions request format.

Restriction of OpenAI Model Parameters

The OpenAI Responses API doesn't support all chat completions parameters. If your request includes unsupported parameters, they're silently dropped by default, and the model call proceeds without them. As a result, parameters such as `frequency_penalty` or `seed` aren't necessarily invoked when you switch to a newer model. If parameters are dropped, you don't receive an error. You can check if any parameters have been dropped by inspecting the `X-Orchestration-Dropped-Model-Params` response header.

Default Behavior: Unsupported Parameters are Dropped

Unsupported parameters that have been silently dropped are included in the response header:

```
X-Orchestration-Dropped-Model-Params: <param1>,<param2>,...
```

The header value is a comma-separated list of the dropped parameter names. It's present on both synchronous and streaming 200 responses whenever at least one parameter is dropped.

Example

`POST /v2/completion`

```
{
  "config": {
    "modules": {
      "prompt_templating": {
        "prompt": {
          "template": [
            {
              "role": "user",
              "content": "Hello"
            }
          ]
        },
        "model": {
          "name": "gpt-6",
          "params": {
            "max_tokens": 100,
            "frequency_penalty": 0.5
          }
        }
      }
    }
  }
}
```

Response (200)

`X-Orchestration-Dropped-Model-Params: frequency_penalty`

```
{
  "request_id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "intermediate_results": { "...": "..." },
  "final_result": { "...": "..." }
}
```

Optional Configuration: Error on Unsupported Parameters

> **⚠ Restriction**
>
> The following is available in Orchestration V2 only.

If you want to receive an error instead of a silent drop, set `drop_unsupported_params: false` on the model object. The service returns HTTP 400 before sending the request to the model.

> **ⓘ Note**
>
> `drop_unsupported_params` belongs on the `model` object, not inside `params` .

Example

`POST /v2/completion`

```
{
  "config": {
    "modules": {
      "prompt_templating": {
```

```
        "prompt": {
          "template": [
            {
              "role": "user",
              "content": "Hello"
            }
          ]
        },
        "model": {
          "name": "gpt-6",
          "params": {
            "max_tokens": 100,
            "frequency_penalty": 0.5
          },
          "drop_unsupported_params": false
        }
      }
    }
  }
}
```

Response (400)

```
{
  "error": {
    "request_id": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "code": 400,
    "message": "Parameter 'frequency_penalty' is not supported by Responses API
model 'gpt-6'. Remove the parameter or set 'drop_unsupported_params: true' in
the model config to silently drop it.",
    "location": "LLM Module"
  }
}
```

Quick Reference

Behavior	`drop_unsupported_params`	HTTP status	Header present
Silent drop (default)	`true` or absent	200	Yes
Hard error	`false`	400	No

Parameters Not Supported by OpenAI Responses API

The following parameters are commonly used:

- `frequency_penalty`
- `logit_bias`
- `logprobs`
- `n`
- `presence_penalty`
- `seed`
- `stop`

The following parameters are legacy or rarely used:

- `audio`

- `function_call`
- `functions`
- `modalities`
- `prediction`
- `verbosity`
- `web_search_options`

> **Note**
>
> `function_call` and `functions` are deprecated predecessors to `tools`. Use `tools` and `tool_choice` for function calling. These parameters are supported by the OpenAI Responses API.

Structured Output

For requests using `response_format` with `"type": "json_schema"`, the `name` field inside `json_schema` is required. For requests that omit it, the model returns a **400** error.

This behavior is enforced by:

- OpenAI Responses API (`gpt-5.3-codex` and all models following `gpt-6`)
- Mistral models (enforced at the model server level regardless of API routing)

> **Note**
>
> Azure-hosted chat completions models (`gpt-5.4`, `gpt-5.5`, and later) previously accepted requests without `name` due to lenient validation by Azure. However, this behavior isn't guaranteed. Always include `name` in your request.

Examples

Correct: (`name` included)

```
"json_schema": {
  "name": "classification_result",
  "schema": {
    "type": "object",
    "properties": {"label": {"type": "string"}
    },
  "required": ["label"]
  }
}
```

Incorrect: (`name` missing)

```
"json_schema": {
  "schema": {
    "type": "object",
    "properties": {"label": {"type": "string"}}
  }
}
```

Related Information

Access the Harmonized API Using the SAP AI SDK for Java

Access the Harmonized API Using the SAP AI SDK for JavaScript

1.4.3.3.1 Provider-Specific Fields

The OpenAI Chat Completion schema, which the Orchestration Service follows, defines a fixed set of fields for each choice. Some providers return information that has no direct equivalent in this schema.

Instead of discarding this information, Orchestration returns it in the optional `provider_specific_fields` field of each choice object. This field contains response data that cannot be harmonized with the OpenAI specification while still preserving access to provider-specific information.

In the following example, the model returns a provider-specific finish reason of `end_turn`, which is specific to Anthropic models. Orchestration normalizes this value to the OpenAI-equivalent value `stop` and preserves the original value in `provider_specific_fields`.

```
{
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Hello!"
      },
      "finish_reason": "stop",
      "provider_specific_fields": {
        "finish_reason": "end_turn"
      }
    }
  ]
}
```

LLM Provider Finish Reason

The `finish_reason` field in the OpenAI schema supports a limited set of values. Orchestration maps provider-specific stop signals to the closest OpenAI-equivalent value.

This mapping can result in a loss of information. A provider such as Anthropic or Vertex AI can use multiple distinct stop signals that map to the same OpenAI value. Provider-specific signal names can also contain diagnostic information that is not preserved in the normalized value.

When the provider-supplied finish reason differs from the mapped OpenAI value, Orchestration preserves the original value in `provider_specific_fields.finish_reason`. This allows you to inspect the exact stop signal returned by the provider while maintaining compatibility with the normalized `finish_reason` field.

1.4.3.4 Embeddings

Semantic embeddings are multidimensional representations of textual information.

The embeddings endpoint is harmonized across different LLMs and can be used alone, or with anonymization of input data.

For more information about supported models, see references to orchestration in Supported Models [page 50].

This feature is only available in orchestration V2.

Send POST request to the endpoint `$ORCHESTRATION_DEPLOYMENT_URL/v2/embeddings` and populate the sample code with the following values:

	Value
$ORCHESTRATION_DEPLOYMENT_URL	The URL of your RUNNING orchestration deployment.
{{token}}	Your authorization token for SAP AI Core
{{resource_group}}	The AI resource group assigned to your account

Minimal Call

The following minimal call shows a model selection and input text.

```
curl --request POST \
  --url $ORCHESTRATION_DEPLOYMENT_URL/v2/embeddings \
  --header 'AI-Resource-Group: {{resource group}}' \
  --header 'Authorization: Bearer {{token}}' \
  --header 'content-type: application/json' \
  --data '{
  "config": {
    "modules": {
      "embeddings": {
        "model": {
          "name": "text-embedding-3-large"
        }
      }
    }
  },
  "input": {
    "text": "Hello, SAP!"
  }
}'
```

Module Calling

The following example includes model parameters and masking configuration. For more information, see Data Masking [page 360] and Model Configuration [page 374].

```
curl --request POST \
  --url $ORCHESTRATION_DEPLOYMENT_URL/v2/embeddings \
  --header 'AI-Resource-Group: {{resource group}}' \
  --header 'Authorization: Bearer {{token}}' \
  --header 'content-type: application/json' \
  --data '{
```

```
  "config": {
    "modules": {
      "embeddings": {
        "model": {
          "name": "text-embedding-3-large",
          "version": "latest",
          "params": {
            "encoding_format": "float",
            "dimensions": 5
          }
        }
      },
      "masking": {
        "providers": [
          {
            "type": "sap_data_privacy_integration",
            "method": "anonymization",
            "entities": [
              {
                "type": "profile-person"
              },
              {
                "type": "profile-email"
              },
              {
                "type": "profile-phone"
              },
              {
                "type": "profile-nationality"
              },
              {
                "type": "profile-university"
              },
              {
                "type": "profile-location"
              },
              {
                "type": "profile-url"
              },
              {
                "type": "profile-gender"
              },
              {
                "type": "profile-credit-card-number"
              }
            ]
          }
        ]
      }
    }
  },
  "input": {
    "text": "Hello, Jake! I had a great meeting with the English stakeholder
from Oxford University, located at 1 Infinite Loop, London. They left their
contacts: manufacturer@gmail.com, or we may call them via +49 1522 3433333 .
Their organization, SAP Public, provided the following details: National
ID 123456789, IBAN DE89370400440532013000, SSN 987-65-4320, passport number
X1234567, and credit card number 4111 1111 1111 1111. Their gender is male,
nationality is British, and their political group is Green Party. For more
info, visit https://sap.com or use the username: jake_user and password:
SuperSecret123!"
  }
}'
```

Harmonization

Orchestration harmonizes the different model provider interfaces under one API. Where supported, the parameter `input_types` defines the mode of embedding generation. Specifying the correct mode can increase accuracy.

> **Note**
>
> For most models this parameter is optional or not supported. For `nvidia--llama-3.2-nv-embedqa-1binput_types` is mandatory.
>
> For more information, see the documentation from the model provider.

Orchestration accepts the following input types:

- `document`: Indexing of document chunks for information retrieval scenarios.
- `query`: Embedding creation of search queries for document search in information retrieval scenarios.
- `text`: Generation of embeddings that are optimized for obtaining similarity scores.

The input type mapping of Orchestration to provider API is as follows:

Google

```
{
    "document": "RETRIEVAL_DOCUMENT",
    "query": "RETRIEVAL_QUERY",
    "text": "SEMANTIC_SIMILARITY"
}
```

NVIDIA `nvidia--llama-3.2-nv-embedqa-1b`

`input_types` is mandatory for this model.

```
{
    "document": "passage",
    "query": "query",
    "type": "query"
}
```

For example:

> **Sample Code**

```
{
    "config": {
        "modules": {
            "embeddings": {
                "model": {
                    "name": "nvidia-llama-3.2-nv-embedqa-1b"
                }
            }
        }
    },
    "input": {
        "text": "My name is Tom. I am applying as a Senior Software Dev. I
work closely with Jerry:",
        "type": "query"
    }
```

```
}
```

Related Information

Access the Embedding using the SAP AI SDK for Java
Access the Embedding using the SAP AI SDK for JavaScript

1.4.3.5 Streaming

Streaming is optional. Where models support streaming, output is generated in chunks, and passed through the modules in the orchestration workflow. This is useful for applications such as chatbots, where interactions with the model happen in real time.

> ⚠ Caution
>
> Streaming can increase the number of calls, increasing consumption and associated costs. For example, if a text of 500 characters generates 5 chunks of 100 characters each, the content filtering module receives 5 calls instead of one.
>
> Streaming can decrease the accuracy of filtering and unmasking, because smaller chunks contain less context. Bench marking and testing with actual use case data is recommended to ensure output quality in relation to chunk size.
>
> For more information, see Metering and Pricing for Generative AI and SAP Note 3505347.

Activate Streaming

Streaming is set to `false` by default, and is part of the `orchestration_config`. To activate streaming, set `stream` to `true`. For example:

V1 Example code

> {·} Sample Code

```
{
  "orchestration_config": {
    "stream": true,
    "module_configurations": {
        ...
     },
     "input_params": {
        ...
     }
  }
}
```

V2 Example code

Sample Code

```
{
    "config": {
        "stream": {
            "enabled": true
        },
        "modules": {
            ...
        }
    },
    "placeholder_values": {
        "input": "sample input param"
    }
}
```

You can configure streaming behavior in the following ways:

- Chunk Size
- Delimiters
- Token Usage

Configure Streaming Using Chunk Size

The `chunk_size` parameter defines the minimum number of characters contained in a single chunk, and takes an integer value.

This means that if a model produces chunks that are larger than the configured `chunk_size`, the Orchestration Service will emit these chunks as is, instead of splitting them into multiple chunks.

This minimizes latency and maximizes the context provided to subsequent tasks like content filtering.

The default value for `chunk_size` is 100.

You can configure the `chunk_size` using the `stream_options` parameter in the `orchestration_config`:

The following example shows a streaming configuration with chunk size of 200 characters.

V1 Example code

Sample Code

```
{
   "orchestration_config": {
      "stream": true,
      "stream_options": {
         "chunk_size": 200
      },
      "module_configurations": {
         ...
         "filtering_module_config": {
            "output": {
               "filters": [
                  ...
               ],
               "stream_options": {
```

```
                    "overlap": 10
                }
            }
        }
        ...
    },
    "input_params": {
        ...
    }
  }
}
```

V2 Example code

Sample Code

```
{
    "config": {
        "stream": {
            "enabled": true,
            "chunk_size": 200
        },
        "modules": {
          ...
            "filtering": {
                "output": {
                    "filters": [
                        ...
                    ],
                    "stream_options": {
                        "overlap": 10
                    }
                }
            }
        },
          ...
        },
        "placeholder_values": {
            "input": "sample input"
        }
    }
}
```

Configure Streaming Using Delimiters

Delimiters can be specified alternatively to, or in addition to chunk size. Streaming configured with the `delimiters` parameter respects the `chunk_size` parameter, but determines the end of a chunk based on the next delimiter in the list, instead of chunk size alone. This means that chunks can be determined in a more meaningful way, for example, to include whole sentences, which may improve output translation, filtering and unmasking results.

You can configure the delimiters using the `stream_options` parameter in the `orchestration_config`:

The following example shows a streaming configuration with delimiters set to common English language sentence endings.

V1 Example code

Sample Code

```
{
  "orchestration_config": {
    "stream": true,
    "stream_options": {
      "delimiters": [".", "!", "?"]
    },
    "module_configurations": {
      ...
    },
    "input_params": {
      ...
    }
  }
}
```

V2 Example code

Sample Code

```
{
   "config": {
     "stream": {
       "enabled": true,
       "delimiters": [".", "!", "?"]
     },
     "modules": {
       ...
     }
   },
   "placeholder_values": {
     "input": "sample input"
   }
 }
```

Include Token Usage in the Streaming Response

By default, streaming responses include token usage information. The `usage` field, which contains the token counts for the request, is added to the streaming response.

You control this behavior with the `include_usage` parameter, which is set within the `stream_options` of the model parameters (`model_params` in V1, and `model.params` in V2).

The `include_usage` parameter takes a boolean value, and the default is `true`. To omit token usage information from the streaming response, set `include_usage` to `false`.

V1 Example code

Sample Code

```
{
   "orchestration_config": {
     "stream": true,
```

```
    "module_configurations": {
      "llm_module_config": {
        "model_name": "gpt-4o",
        "model_params": {
          "stream_options": {
            "include_usage": false
          }
        }
      },
      ...
    },
    "input_params": {
      ...
    }
  }
}
```

V2 Example code

Sample Code

```
{
   "config": {
     "stream": {
       "enabled": true
     },
     "modules": {
       "prompt_templating": {
         "model": {
           "name": "gpt-4o",
           "params": {
             "stream_options": {
               "include_usage": false
             }
           }
         }
       },
       ...
     }
   },
   "placeholder_values": {
     "input": "sample input"
   }
}
```

Module Specific Streaming Behavior

The following orchestration modules use content passed from streaming:

- Data Masking (unmasking pseudonymization only)
- Output filtering

Data Masking

For the data masking module, the full `MASKED_ENTITY_DIGIT` tag is needed for unmasking. When a tag is split by chunking, the entire tag is moved to the following chunk, changing the chunk size.

For example, two chunks containing:

Sample Code

```
Yesterday, I spent time with MASKED_PER
SON_1 discussing the changes
```

Would become:

Sample Code

```
Yesterday, I spent time with
MASKED_PERSON_1 discussing the changes
```

This may affect behavior of future modules that are executed after unmasking.

Output filtering

The `overlap` parameter defines the number of characters at the end of a chunk, that are repeated at the beginning of the next chunk. It takes an integer value. The default value is 0.

Increasing the context window improves output filtering results, but increasing chunk size can affect streaming experience negatively. Overlap repeats a specified amount of text across two adjacent chunks, creating a larger context window, without increasing chunk size.

For example:

Sample Code

```
"This is the text from the 1st chunk."
"This is the text from the 2nd chunk"
```

Would become:

Sample Code

```
"This is the text from the 1st chunk."
"the 1st chunk. This is the text from the 2nd chunk"
```

You can configure the `overlap` parameter using the `output stream_options overlap parameter` in the `filtering_module_config`.

The following example shows a streaming configuration with an overlap of 10 characters.

Sample Code

```
{
   "orchestration_config": {
      "stream": true,
      "stream_options": {
         "chunk_size": 200
      },
      "module_configurations": {
         ...
         "filtering_module_config": {
            "output": {
```

```
                "filters": [
                   ...
                ],
                "stream_options": {
                   "overlap": 10
                }
             }
          }
          ...
      },
      "input_params": {
         ...
      }
   }
}
```

> **Note**
>
> If you enable Azure Content Safety Protected Material Code detection (`protected_material_code`) in a streaming request, both `chunk_size` and the output filtering `overlap` fields must be set to values of at least 111. Otherwise the request is rejected with a HTTP `400 Bad Request` response. This ensures the content sent for scanning always meets the service's minimum character requirement. For more information, see Content Filtering [page 560].
>
> Setting both values to at least 111 doesn't guarantee that every chunk clears the minimum. If you also configure `delimiters`, a chunk can be split at a delimiter and fall below 111 characters, which is rejected mid-stream with a `400`-coded error event.

Streaming Response Format

Streaming responses follow the conventions used by OpenAI models and are sent as Server-Sent Events (SSEs).

The first chunk always contains the module results of all modules executed before calling the LLM.

The LLM's output starts from the second chunk, and each chunk produced by orchestration sends a data event followed by a newline.

Successful responses are terminated with `[DONE]`. If an error occurs, orchestration sends a data event containing an error object as payload, and doesn't terminate with `[DONE]`.

V1 Example for a Successful Response

The first data event contains the module result of modules that are run before the LLM access. This example shows the templating module result, and an empty orchestration result:

> **Output Code**

```
data: {
  "request_id": "...",
       "module_results": {
         "templating": [
           {
             "role": "user",
             "content": "Create 3 paraphrases of I love you."}
         ]
```

```
        },
        "orchestration_result": {
          "id": "",
          "object": "",
          "created": 0,
          "model": "",
          "system_fingerprint": "",
          "choices": [
            {
              "index": 0,
              "delta": {
                "role": "",
                "content": ""
              },
              "finish_reason": ""
            }
          ]
        }
      }
```

The second and following data events contain module results from the LLM and following modules. This example shows results for LLM access and orchestration. The orchestration result is given in two chunks.

First chunk:

Output Code

```
data: {
  "request_id": "...",
  "module_results": {
    "llm":
    {
      ...
    }
  },
  "orchestration_result": {
    "id": "...",
    "object": "chat.completion.chunk",
    "created": 1728043660,
    "model": "<ModelName",
    "choices": [
      {
        "index": 0,
        "delta": {
          "role": "assistant",
          "content": "1. My affection for you knows no bounds.\n2. You hold a
special place in my heart.\n3. You mean the wo"
        },
        "finish_reason": ""
      }
    ]
  }
}
```

Second chunk:

Output Code

```
data: {
  "request_id": "163ea321-8f4f-4d00-9b34-7603f1499c24",
  "module_results": {
    "llm": {
      ...
    }
```

```
  },
  "orchestration_result": {
    "id": "...",
    "object": "chat.completion.chunk",
    "created": 1728043660,
    "model": "<ModelName",
    "choices": [
      {
        "index": 0,
        "delta": {
          "role": "assistant",
          "content": "rld to me."
        }, "finish_reason": "stop"
      }
    ]
  }
}
```

The final data-event contains `[DONE]`:

Output Code

```
data: [DONE]
```

Example for a V1 Error Response

If an error occurs, orchestration sends a single data event containing the error object, and the stream ends without a `[DONE]` event. The error object contains the `request_id`, the error `code`, a `message`, the `location` where the error occurred, and the `module_results` produced before the error:

Sample Code

```
data: {
  "request_id": "...",
  "code": 500,
  "message": "...",
  "location": "...",
  "module_results": {
    ...
  }
}
```

V2 Example for a Successful Response

The `intermediate_results` of first data event contains the module result of modules that are run before the LLM access. This example shows the templating module result, and an empty `final_result` which is orchestration result:

Output Code

```
data: {
  "request_id": "...",
       "intermediate_results": {
         "templating": [
           {
             "role": "user",
             "content": "Create 3 paraphrases of I love you."}
         ]
       },
```

```
      "final_result": {
        "id": "",
        "object": "",
        "created": 0,
        "model": "",
        "system_fingerprint": "",
        "choices": [
          {
            "index": 0,
            "delta": {
              "role": "",
              "content": ""
            },
            "finish_reason": ""
          }
        ]
      }
    }
```

The `intermediate_results` of second and following data events contain module results from the LLM and following modules. This example shows results for LLM access and orchestration. The orchestration result is given in two chunks.

First chunk:

Output Code

```
data: {
  "request_id": "...",
  "intermediate_results": {
    "llm":
    {
      ...
    }
  },
  "final_result": {
    "id": "...",
    "object": "chat.completion.chunk",
    "created": 1728043660,
    "model": "<ModelName",
    "choices": [
      {
        "index": 0,
        "delta": {
          "role": "assistant",
          "content": "1. My affection for you knows no bounds.\n2. You hold a
special place in my heart.\n3. You mean the wo"
        },
        "finish_reason": ""
      }
    ]
  }
}
```

Second chunk:

Output Code

```
data: {
  "request_id": "163ea321-8f4f-4d00-9b34-7603f1499c24",
  "intermediate_results": {
    "llm": {
      ...
```

```
    }
  },
  "final_result": {
    "id": "...",
    "object": "chat.completion.chunk",
    "created": 1728043660,
    "model": "<ModelName>",
    "choices": [
      {
        "index": 0,
        "delta": {
          "role": "assistant",
          "content": "rld to me."
        }, "finish_reason": "stop"
      }
    ]
  }
}
```

The final data-event contains `[DONE]`:

Output Code

```
data: [DONE]
```

V2 Example for an Error Response

If an error occurs, orchestration sends a single data event containing the error object wrapped in an `error` field, and the stream ends without a `[DONE]` event. The error object contains the `request_id`, the error `code`, a `message`, the `location` where the error occurred, and the `intermediate_results` produced before the error:

Output Code

```
data: {
  "error": {
    "request_id": "...",
    "code": 500,
    "message": "...",
    "location": "...",
    "intermediate_results": {
      ...
    }
  }
}
```

Related Information

Access the Streaming using the SAP AI SDK for Java

Access the Streaming using the SAP AI SDK for JavaScript

1.4.3.6 Model Restriction

When creating a deployment for orchestration, you can restrict the choice of models using an allow or deny list. You can use this to implement internal standards, for example where only certain LLMs are approved for use.

To restrict model access, set the `parametersBindings` to contain `modelFilterList` and `modelFilterListType`.

- **modelFilterList**: a list of `modelNames` and `modelVersions` to be considered.
  If `modelVersions` is not defined, all versions of the given model are considered.
- **modelFilterListType**: a value that controls how `modelFilterList` should be interpreted.
  - `deny`: excludes the models and defined versions from use within orchestration.
  - `allow`: only allows the models and defined versions in the modelFilterList to be used within orchestration.

Sample Code

```
{
  "name": "yourNameChoice",
  "executableId": "orchestration",
  "scenarioId": "orchestration",
  "versionId": "0.0.1",
  "parameterBindings": [
    {
      "key": "modelFilterList",
      "value": "[{\"modelName\": \"gpt-4o-mini\", \"modelVersions\":
[\"0613\", \"0301\"]}, {\"modelName\": \"gemini-1.5-pro\", \"modelVersions\":
[\"001\"]}]"
    },
    {
      "key": "modelFilterListType",
      "value": "deny"
    }
  ]
}
```

1.4.3.7 Reasoning

Note

`reasoning_content` is only available in Orchestration V2.

Orchestration supports models with reasoning capabilities. Reasoning is harmonized across all supported providers under a unified API specification, independent of the underlying model's native format.

Enabling Reasoning

To enable reasoning, set the `reasoning_effort` parameter in the model configuration. When set, the model is instructed to perform reasoning before producing a response.

The possible values are:

- `"minimal"`
- `"low"`
- `"medium"`
- `"high"`
- `"xhigh"`
- `"none"` (disables reasoning where supported)

> **Note**
>
> Not all models support the full range of configuration options. Supported values vary by model. If you're unsure which value to use, start with `low`, `medium`, or `high`, as these values are widely supported. When possible, orchestration maps unsupported values to the closest supported equivalent.

Reasoning Content

When a model performs reasoning, orchestration returns the progress in the `reasoning_content` field.

Response Format

> **Output Code**

```
{
  "choices": [
    {
      "message": {
        "role": "assistant",
        "content": "The answer is 42.",
        "reasoning_content": [
          {
            "content": "Let me think through this step by step...",
            "signature": "CiUBjz1rX..."
          }
        ]
      }
    }
  ]
}
```

Each reasoning block has two fields:

Field	Description
`content`	The reasoning text produced by the model.

Field	Description
`signature`	An opaque string required to replay reasoning in follow-up requests. May be an empty, as not all models return a signature.

> **Note**
>
> Not all reasoning models expose their internal reasoning processes. Reasoning content is returned whenever the model provides it.

Streaming

In streaming responses, reasoning content is spread across multiple chunks. The `signature` may appear in a different chunk than the `content`, depending on the model. Always accumulate all `reasoning_content` blocks across all chunks before using them in a follow-up request.

```
{
  "choices": [
    {
      "delta": {
        "role": "assistant",
        "content": "",
        "reasoning_content": [
          {
            "content": "Let me think through this step by step...",
            "signature": ""
          }
        ]
      }
    }
  ]
}
```

A later chunk may carry the signature with empty content:

```
{
  "choices": [
    {
      "delta": {
        "role": "assistant",
        "content": "",
        "reasoning_content": [
          {
            "content": "",
            "signature": "CiUBjz1rX..."
          }
        ]
      }
    }
  ]
}
```

Multi-Turn Conversations

When continuing a conversation, always include the full list `reasoning_content` blocks from the previous assistant message in messages_history **exactly as returned by the API**.

- **Do not modify** the `content` or `signature` fields.
- **Do not omit** reasoning blocks, even if they have an empty `content` or `signature`.
- **Do not merge** multiple blocks into one.

Returning reasoning content generally improves model accuracy and is recommended for all providers.

> **Note**
>
> For Anthropic models, it is **strictly required** when tool calls are present in the same assistant turn.

> **Sample Code**

```
{
  "config": {
    "modules": {
      "prompt_templating": {
        "prompt": {
          "template": [
            { "role": "user", "content": "Are you sure?" }
          ]
        },
        "model": {
          "name": "anthropic--claude-4-sonnet",
          "params": { "reasoning_effort": "medium" }
        }
      }
    }
  },
  "messages_history": [
    { "role": "user", "content": "What is 6 times 7?" },
    {
      "role": "assistant",
      "content": "The answer is 42.",
      "reasoning_content": [
        {
          "content": "6 times 7 equals 42.",
          "signature": "CiUBjz1rX..."
        }
      ]
    }
  ]
}
```

1.4.3.8 Structured Output

You can use structured outputs to ensure that model generated outputs match schemas that you define.

Structured outputs exist in the following forms:

- Function calling, such as for tool usage

Function calling is helpful for agentic use cases where the LLM determines which tool to use out of a provided selection of tools. The LLM determines which tool is best for the use case, but the tool needs to be called separately in the application.

Sample Code

```
"templating_module_config": {
    "template": [
        {
            "role": "system",
            "content": "You are a helpful assistant. You help users do
math by calling the abs function."
        },
        {
            "role": "user",
            "content": "Whats the absolute value of 3?"
        }
    ],
    "tools": [
        {
            "type": "function",
            "function": {
                "name": "abs",
                "description": "Calculate the absolute value of x.",
                "strict": true,
                "parameters": {
                    "type": "object",
                    "properties": {
                        "x": {
                            "type": "integer"
                        }
                    },
                    "required": [
                        "x"
                    ],
                    "additionalProperties": false
                }
            }
        }
    ]
}
```

- JSON response format
  JSON formats can be consumed in an application

```
"templating_module_config": {
      "template": [
          {
              "role": "system",
              "content": "You are a language translator."
          },
          {
              "role": "user",
              "content": "Whats 'Apple' in German?"
          }
      ],
      "response_format": {
          "type": "json_schema",
          "json_schema": {
              "name": "translation_response",
              "strict": true,
              "schema": {
                  "type": "object",
                  "properties": {
                      "language": {
```

```
                            "type": "string"
                        },
                        "translation": {
                            "type": "string"
                        }
                    },
                    "required": [
                        "language",
                        "translation"
                    ],
                    "additionalProperties": false
                }
            }
        }
    }
```

Orchestration harmonizes tool usage and JSON response format across different models. For more information on which models support these features, see the documentation from the respective model provider.

For more information on structured outputs, see Open AI Structured Outputs and Open AI Structured Output Guide .

1.4.3.9 Error Handling

If an error occurs, the response will contain an error code and message. The following examples show a request that is missing a parameter in the template:

Example: V1

```
curl --request POST $ORCH_DEPLOYMENT_URL/completion \
    --header 'content-type: application/json' \
    --header "Authorization: Bearer $TOKEN" \
    --header "AI-Resource-Group: $RESOURCE_GROUP" \
    --data-raw '{
    "orchestration_config": {
      "module_configurations": {
        "templating_module_config": {
          "template": [
            {
              "role": "user",
              "content": "Create {{?number}} paraphrases of {{?phrase}}"
            }
          ]
        },
        "llm_module_config": {
          "model_name": "<ModelName>",
          "model_params": {
            "max_tokens": 300
          }
        }
      }
    },
    "input_params": {
      "number": "3"
    }
  }
```

Example V2

```
curl --request POST $ORCH_DEPLOYMENT_URL/v2/completion \
    --header 'content-type: application/json' \
    --header "Authorization: Bearer $TOKEN" \
    --header "AI-Resource-Group: $RESOURCE_GROUP" \
    --data-raw '{
      "config": {
        "modules": {
          "prompt_templating": {
            "prompt": {
              "template": [
                {
                  "role": "user",
                  "content": "Create {{?number}} paraphrases of {{?phrase}}"
                }
              ]
            },
            "model": {
              "name": "<model-name>",
              "params": {
                "max_completion_tokens": 300
               }
            }
          }
        }
      },
      "placeholder_values": { } // no placeholder_values supplied
    }
```

In this case, the response from the orchestration service contains an error code in the `code` field, a `message`, and a `location` indicating which orchestration module encountered the error.

Output: V1

Output Code

```
{
  "request_id": "3e988846-1360-4a4a-a7ad-e77b85057321",
  "code": 400,
  "message": "Missing required parameters: ['phrase']",
  "location": "Module: Templating",
  "module_results": {}
}
```

Output: V2

Output Code

```
{
  "error": {
    "request_id": "3e988846-1360-4a4a-a7ad-e77b85057321",
    "code": 400,
    "message": "400 - Input Parameters: Error validating parameters. Unused
 parameters: ['number', 'phrase'].",
    "location": "Input Parameters",
    "intermediate_results": {
      "templating": [
        {
          "content": "Create {{?number}} paraphrases of {{?phrase}}",
          "role": "user"
        }
      ]
```

```
    }
  }
}
```

1.4.3.10 Orchestration with Fallbacks

You can define multiple sets of module configurations for completions in your orchestration workflow to incorporate fallbacks in case of processing failure, such as unsupported models in the deployed region. Requests are processed using the configuration with the highest preference. If processing fails, the next preferred configuration is automatically used for processing. Preference order follows the sequence of the list: the first item has the highest priority, and the last item has the lowest priority.

This feature is only available in orchestration V2.

Criteria for switching to fallback configurations

For streaming and non-streaming requests: orchestration switches to fallback configurations when the specified model in the higher-preference instance is not supported in the deployed region.

For non-streaming requests: orchestration may also switch to the next fallback if it encounters certain errors during processing. This allows orchestration to leverage fallback configurations and to provide successful responses in case of transient errors, such as, temporary unavailability of a model or service. For streaming requests, switching to a fallback configuration is not straightforward once the stream has started.

Orchestration will only switch to the next defined fallback configuration if it encounters specific errors during processing. For all other errors, orchestration will fail the request and return the corresponding error in the response.

The fallback configuration will be used in the following scenarios:

- For streaming requests:
  - If specified model is not supported in the deployed region or environment
- For non-streaming requests:
  - If specified model is not supported in the deployed region or environment
  - If orchestration service encounters any of the following errors during processing:
    - Client errors: `429 Too Many Requests` and `408 Request Timeout`
    - All server errors `5xx`

Example

The following example shows a request with: the specified `modules` list indicating that `gpt-4o` is preferred as the primary model, with `mistralai—mistral-small-instruct` as the fallback option. If `gpt-4o` is unsupported in the deployed region, orchestration automatically switches to the second preference. Module

configurations can vary between preferences to suit different use cases. In this case, the fallback configuration omits the filtering module.

Sample Code

```
curl --request POST $ORCH_DEPLOYMENT_URL/v2/completion \
--header 'content-type: application/json' \
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP" \
--data-raw '{
    "config": {
        "modules": [
            { //1st preference
               "prompt_templating": {
                    "prompt": {
                        "template": [
                            {
                                "role": "system",
                                "content": "You are a language translation
model that translates English to German."
                            },
                            {
                                "role": "user",
                                "content": "{{ ?input }}"
                            }
                        ]
                    },
                    "model": {
                        "name": "gpt-4o",
                        "params": {
                            "max_tokens": 300,
                            "temperature": 2
                        }
                    }
                },
                "filtering": {
                    "input": {
                        "filters": [
                            {
                                "type": "azure_content_safety",
                                "config": {
                                    "prompt_shield": "true"
                                }
                            }
                        ]
                    }
                }
            },
            { //2nd preference
                "prompt_templating": {
                    "prompt": {
                        "template": [
                            {
                                "role": "system",
                                "content": "You are a language translation
model that translates English to German."
                            },
                            {
                                "role": "user",
                                "content": "{{ ?input }}"
                            }
                        ]
                    },
                    "model": {
                        "name": "mistralai--mistral-small-instruct",
                        "params": {
                            "max_tokens": 300,
```

```
                            "temperature": 2
                        }
                    }
                }
            }
        ]
    },
    "placeholder_values": {
        "input": "Reply with Orchestration Service is working! in German"
    }
}'
```

The successful response below was processed using the fallback preference `mistralai—mistral-small-instruct`. The `intermediate_failures` field indicates why the highest preference model was skipped.

Output Code

```
{
    "request_id": "f380e708-cd64-9b87-b36b-03b53a678f59",
    "intermediate_results": {
        "templating": [
            {
                "content": "Reply with Orchestration Service is working! in
German",
                "role": "user"
            }
        ],
        "llm": {
            "id": "chatcmpl-0d66dbb6-6a3b-9aa5-a2c3-4f4c9e608051",
            "object": "chat.completion",
            "created": 1761909762,
            "model": "mistralai--mistral-small-instruct",
            "choices": [
                {
                    "index": 0,
                    "message": {
                        "role": "assistant",
                        "content": "Sure! Here is the
translation:\n\n\"Orchestration Service läuft!\"\n\nThis means
\"Orchestration Service is working!\" in German."
                    },
                    "finish_reason": "stop"
                }
            ],
            "usage": {
                "completion_tokens": 30,
                "prompt_tokens": 14,
                "total_tokens": 44
            }
        }
    },
    "final_result": {
        "id": "chatcmpl-0d66dbb6-6a3b-9aa5-a2c3-4f4c9e608051",
        "object": "chat.completion",
        "created": 1761909762,
        "model": "mistralai--mistral-small-instruct",
        "choices": [
            {
                "index": 0,
                "message": {
                    "role": "assistant",
                    "content": "Sure! Here is the
translation:\n\n\"Orchestration Service läuft!\"\n\nThis means
\"Orchestration Service is working!\" in German."
                },
                "finish_reason": "stop"
```

```
            }
        ],
        "usage": {
            "completion_tokens": 30,
            "prompt_tokens": 14,
            "total_tokens": 44
        }
    },
    "intermediate_failures": [
        {
            "code": 400,
            "message": "Model gpt-4o not supported.",
            "location": "Request Body",
            "headers": {
                "Content-Type": "application/json"
            }
        }
    ]
}
```

Error Response

If orchestration is unable to process the request using any of the provided fallback module configurations, it responds with a list of errors corresponding to each failed attempt. The overall response code matches the last failed attempt, and the error list follows the defined preference order.

Output Code

```
{
    "error": [// 1st preference        {
            "request_id": "req-id",
            "code": 400,
            "message": "400 - Request Body: Model gpt-4o not supported.",
            "location": "Request Body",
            "headers": {
                "Content-Type": "application/json"
            }
        },
        {// 2nd preference
            "request_id": "req-id",
            "code": 429,
            "message": "429 - Too many requests.",
            "location": "LLM",
            "headers": {
                "Content-Type": "application/json",
                "x-retry-after": "30"
            }
        }
    ]
}
```

1.4.3.11 Orchestration Tutorials

Find your use case with orchestration capabilities.

For a complete list of tutorials on orchestration, see Generative AI with SAP AI Core - Orchestration.

Related Information

Libraries and SDKs

1.4.4 Tabular AI

Prerequisites

- You have an SAP AI Core service instance and service key. For more information, see SAP AI Core Initial Setup Documentation.
- You're using the `extended` service plan. For more information, see Service Plans and Update a Service Plan.
- You've chosen the model that you want to make a deployment for. For more information, see Choose a Model [page 192].

Introduction

Structured enterprise data, represented as rows and columns, underpins business processes across SAP systems. Tabular AI enables artificial intelligence capabilities directly on this structured data to generate predictions and insights.

While large language models focus on unstructured text, most enterprise business value resides in relational data. Tabular AI addresses this gap by operating specifically on tabular data sources.

SAP RPT-1 is a foundation model designed for tabular data. It uses in-context learning and does not require training on a specific dataset. You provide representative rows as context at inference time, and the model generates predictions immediately.

Tabular AI extends beyond the model itself. It provides the orchestration layer that integrates predictive models into enterprise business processes by managing context, metadata, and execution complexity.

Core Principles

Tabular AI is built around the following core principles:

- A unified interface for existing and future predictive models.
- Integrated tooling that manages context selection automatically.
- End-to-end workflow support across design time and runtime.

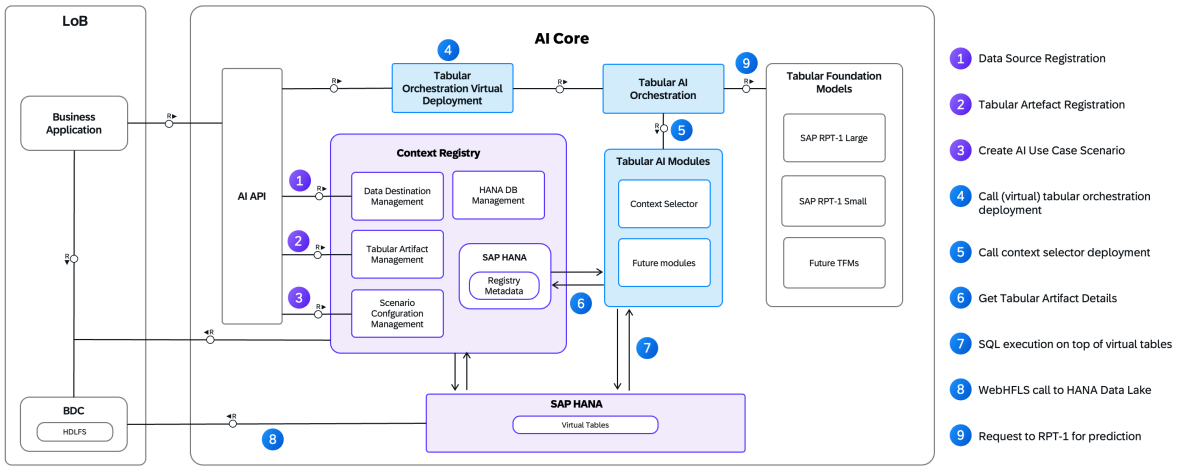
The Context Registry is a central component of this architecture. It manages data source connectivity, schema metadata, and scenario configuration. This abstraction ensures that application developers do not need to manage these details directly.

Architecture Overview

The Tabular AI architecture separates design-time configuration from runtime execution. This separation supports scalability across multiple teams and use cases.

Design-time activities define how data is accessed, interpreted, and used for prediction. Runtime execution focuses on invoking predictions using the predefined configuration.

Architecture Overview



Design-Time Configuration

During design time, you define the data source, tabular artifact, and scenario configuration. These elements remain unchanged during runtime prediction requests.

After design-time configuration is complete, the system is ready to serve predictions without further setup.

For more information, see Design-Time Setup [page 605].

Runtime Execution

At runtime, an application invokes the Tabular AI orchestration layer to request a prediction. The system automatically retrieves relevant context, assembles the inference request, and routes it to the appropriate model.

The application interacts only with the invocation interface, while the platform manages all internal steps required to generate the prediction result.

To setup your tabular AI runtime, create a tabular AI deployment. For more information, see Create a Tabular AI Deployment [page 629].

After setup, you can consume tabular AI. For more information, see Consume Tabular Orchestration [page 631].

1.4.4.1 Design-Time Setup

Design-time configuration prepares Tabular AI for runtime prediction. During this setup, you define how Tabular AI accesses data, interprets schema metadata, and selects relevant context rows for prediction.

Design-time setup is a one-time process that must be completed before you make predictions. It consists of the following steps:

1. Connect Tabular AI to your storage back end.
    1. Test the connection. No configuration is saved during the test.
    2. Create the destination after the connection status changes to OK.

    For more information, see Data Destination [page 607].
2. Onboard a specific file and its schema.
    1. Submit the artifact creation request.
    2. Poll until the status changes to `READY`.
    3. Optionally, verify the data.

    For more information, see Tabular Artifact [page 616].
3. Bind the tabular artifact to a context selection strategy. For more information, see Scenario Configuration [page 622].

Common Conventions

Make sure that you've set the following **headers**:

Key	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps

Make sure you have set the following variable:

- **$AI_API_URL**: The base URL of your SAP AI Core environment. You can set this URL as an environment variable.

Pagination Query Parameters

The following parameters are supported:

Parameter	Description	Type
`$top`	Maximum number of results to return	Full result set
`$skip`	Number of results to skip	integer (≥ 0)
$count	When `true`, `count` reflects the total server-side record count	boolean

Naming Rules

Resource	Pattern	Max Length
Data Destination	`^[a-z0-9]([-a-z0-9]*[a-z0-9])?$`	127 chars
Tabular Artifact	`^[a-z0-9]([-a-z0-9]*[a-z0-9])?$`	80 chars
Scenario Configuration	`^[a-z0-9]([-a-z0-9]*[a-z0-9])?$`	127 chars

Entries must consist only of lowercase letters, digits, and hyphens. They must start and end with an alphanumeric character.

Label Format

Field	Pattern	Max Length
`key`	Must start with `ext.ai.sap.com/`	63 chars
`value`	Alphanumeric character with hyphen (-), underscore (_), period (.), comma (,), or slash (/)	63 chars

When you PATCH labels, they're replaced in full, not merged with existing ones.

Error Response Format

```
{
  "error": {
    "code": "RESOURCE_NOT_FOUND",
    "message": "Data destination 'my-dd' not found",
    "requestId": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
    "target": "/api/v1/tcr/dataDestinations/my-dd"
  }
}
```

1.4.4.1.1 Data Destination

A Data Destination represents a connection to an external storage system. It encapsulates connection credentials so that downstream components never handle them directly.

Every Tabular Artifact must reference a Data Destination.

The following types are supported:

Type	Storage System
HDL	SAP HANA Data Lake Files
S3	Amazon S3
GCS	Google Cloud Storage
AZURE	Azure Blob Storage (ADLS Gen2)

Create a Data Destination

> **Note**
>
> Data destination creation is asynchronous. The service immediately accepts the request and returns `202 Accepted` while credential validation and secret storage are processed in the background. Credentials are stored securely and are not returned in `GET` or `LIST` responses.
>
> Before creating a tabular artifact, retrieve the data destination and verify that its status is `ACTIVE`. If creation fails, the destination status is `ERROR`, and the `errorMessage` field provides details about the failure. To retry the operation, send another `PUT` request with the same destination name.
>
> Specify the destination name in the request path, not in the request body.

Query Parameters

Parameter	Type	Description	Applicable To
`namedSp`	boolean	When set to `true`, returns HDL subject patterns in the `namedSubjectPatterns` object in addition to `subjectPatterns`.	HDL data destinations (PUT endpoints)

Use the `subjectPatterns` values returned in the HDL response to configure mTLS trust on the SAP HANA Data Lake side.

Use the following API to register your data destination by sending a PUT request to the endpoint: `$AI_API_URL/v2/tcr/dataDestinations/<name>`

Endpoint

Method	Path	Description
PUT	`/dataDestinations/{name}`	Create or retry a data destination asynchronously

HDL

Populate the following with your details:

```
curl PUT -X "$AI_API_URL/v2/tcr/dataDestinations/hdl-production" \
  --header "AI-Resource-Group: $RESOURCE_GROUP_ID" \
  --header "Authorization: Bearer $TOKEN" \
  --header "Content-Type: application/json" \
  --data '{
  "type": "HDL",
  "description": "HDL data destination for payment prediction use case",
  "adapterType": "File",
  "config": {
    "host": "<your-container-id>.files.hdl.<region>.hanacloud.ondemand.com"
  }
}'
```

> **Note**
>
> The data destination name must contain only lowercase alphanumeric characters and hyphens. The host value corresponds to the HANA Data Lake container endpoint available in the SAP BTP subaccount.

Results

A successful request returns HTTP status code `202 Accepted` along with the generated mTLS subject patterns.

> **Output Code**

```
{
  "name": "hdl-production",
  "subjectPatterns": [
    "sap.ai.core:resourceGroup:my-resource-group:dataDestination:hdl-
production"
  ]
}
```

AWS

Populate the following with your details:

```
curl PUT -X "$AI_API_URL/v2/tcr/dataDestinations/s3-training-data" \
  --header "AI-Resource-Group: $RESOURCE_GROUP_ID" \
  --header "Authorization: Bearer $TOKEN" \
  --header "Content-Type: application/json" \
  --data '{
    "type": "S3",
```

```
    "description": "S3 bucket for training datasets",
    "config": {
      "bucket": "my-ml-training-bucket",
      "region": "eu-central-1",
      "access_key_id": "AKIAIOSFODNN7EXAMPLE",
      "secret_access_key": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
    }
  }'
```

Output Code

```
Response — 202 Accepted:
{
  "name": "s3-training-data"
}
```

GCS

Populate the following with your details.

The `base64_encoded_private_key_data` is the full service account JSON key, base64-encoded, which is the same value provided by the BTP Object Store service key:

```
curl PUT -X "$AI_API_URL/v2/tcr/dataDestinations/gcs-analytics" \
  --header "AI-Resource-Group: $RESOURCE_GROUP_ID" \
  --header "Authorization: Bearer $TOKEN" \
  --header "Content-Type: application/json" \
  --data '{
    "type": "GCS",
    "config": {
      "bucket": "my-gcs-analytics-bucket",
      "base64_encoded_private_key_data":
"eyJjbGllbnRfZW1haWwiOiJzZXJ2aWNlQGV4YW1wbGUuaWFtLmdzZXJ2aWNlYWNjb3VudC5jb20iLCJw
cml2YXRlX2tleSI6Ii0tLS0tQkVHSU4gUFJJVkFURSBLRVktLS0tLVxuLi4uXG4tLS0tLUVORCBQUklWQ
VRFIEtFWS0tLS0tIn0="
    }
  }'
```

Output Code

```
Response — 202 Accepted:
{
  "name": "gcs-analytics"
}
```

Azure

Populate the following with your details:

```
curl PUT -X "$AI_API_URL/v2/tcr/dataDestinations/azure-blob-store" \
  --header "AI-Resource-Group: $RESOURCE_GROUP_ID" \
  --header "Authorization: Bearer $TOKEN" \
```

```
  --header "Content-Type: application/json" \
  --data '{
    "type": "AZURE",
    "config": {
      "account_name": "mystorageaccount",
      "container_uri": "https://mystorageaccount.z2.blob.storage.azure.net/
mycontainer",
      "sas_token":
"sv=2023-01-03&ss=b&srt=sco&sp=rl&se=2025-12-31T00:00:00Z&st=2024-01-01T00:00:00Z
&spr=https&sig=EXAMPLE_SIG"
    }
  }'
```

Output Code

```
Response — 202 Accepted:
{
  "name": "azure-blob-store"
}
```

Error Responses

The results may include the following error responses:

Status	When
400	Missing required fields, name pattern violation, or invalid `host` format
409	A data destination with this name already exists in the resource group
500	Connectivity validation failed or internal error

Next Step

Note

Creating a data destination is an asynchronous operation. The service immediately accepts the request and returns `202 Accepted`, while credential validation and secret storage are processed in the background. Credentials are stored securely and aren't included in GET or LIST responses.

Before creating a tabular artifact, poll `GET /dataDestinations/{name}` until the data destination status changes to `ACTIVE`.

If the operation fails, the data destination status changes to `ERROR`. The response includes an `errorMessage` field that describes the reason for the failure.

To retry the operation, send another PUT request with the same data destination name.

For more information about retrieving a data destination status, see Retrieve Destination.

1.4.4.1.1.1 Manage Data Destinations

The service provides APIs to **retrieve**, **list**, **update**, and **search** data destinations.

Endpoints

Method	Path	Description
GET	`/dataDestinations`	List all data destinations
GET	`/dataDestinations/{name}`	Get details (credentials excluded)
PATCH	`/dataDestinations/{name}`	Update labels, description, or credentials
DELETE	`/dataDestinations/{name}`	Mark for asynchronous deletion
POST	`/dataDestinations/search`	Search by label key-value pairs
POST	`/dataDestinations/validate`	Test connectivity before creating
POST	`/dataDestinations/{name}/validate`	Re-verify an existing destination

Query Parameters

Parameter	Type	Description	Applicable To
`namedSp`	boolean	When set to `true`, returns HDL subject patterns in the `namedSubjectPatterns` object in addition to `subjectPatterns`.	HDL data destinations (GET endpoints)

List Data Destinations

Send a GET request to list all data destinations:

```
curl GET "$AI_API_URL/v2/tcr/dataDestinations?\$top=10&\$count=true" \
  --header "AI-Resource-Group: $RESOURCE_GROUP_ID" \
  --header "Authorization: Bearer $TOKEN"
```

Successful responses return code **200** and item details.

```
{
  "count": 2,
  "resources": [
    {
      "name": "hdl-production",
      "type": "HDL",
      "description": "Production HANA Data Lake file container",
      "adapterType": "File",
      "status": "ACTIVE",
      "labels": [
        { "key": "ext.ai.sap.com/environment", "value": "production" }
      ],
```

```
      "createdAt": "2024-03-10T09:00:00.000Z",
      "updatedAt": "2024-03-10T09:00:00.000Z"
    },
    {
      "name": "s3-training-data",
      "type": "S3",
      "description": "S3 bucket for training datasets",
      "adapterType": "File",
      "status": "ACTIVE",
      "labels": [],
      "createdAt": "2024-03-11T11:30:00.000Z",
      "updatedAt": "2024-03-11T11:30:00.000Z"
    }
  ]
}
  ]
}
```

Retrieve a Data Destination

Retrieve details of a specific data destination by sending a GET request to the following endpoint:

```
curl GET "$AI_API_URL/v2/tcr/dataDestinations/hdl-production" \
  --header "AI-Resource-Group: $RESOURCE_GROUP_ID" \
  --header "Authorization: Bearer $TOKEN"
```

Successful responses return code **200** and item details.

The destination status can be ACTIVE, PROCESSING, or ERROR.

```
{
  "name": "hdl-production",
  "type": "HDL",
  "description": "Production HANA Data Lake file container",
  "adapterType": "File",
  "status": "ACTIVE",
  "labels": [
    { "key": "ext.ai.sap.com/environment", "value": "production" }
  ],
  "createdAt": "2024-03-10T09:00:00.000Z",
  "updatedAt": "2024-03-10T09:00:00.000Z"
}
```

```
{
  "name": "hdl-production",
  "type": "HDL",
  "description": "Production HANA Data Lake file container",
  "adapterType": "File",
  "status": "PROCESSING",
  "labels": [
    { "key": "ext.ai.sap.com/environment", "value": "production" }
  ],
  "createdAt": "2024-03-10T09:00:00.000Z",
  "updatedAt": "2024-03-10T09:00:00.000Z"
}
```

```
{
  "name": "hdl-production",
  "type": "HDL",
  "description": "Production HANA Data Lake file container",
```

```
  "adapterType": "File",
  "status": "ERROR",
  "errorMessage": "Failed to validate credentials: Access denied to HDL
endpoint",
  "labels": [
    { "key": "ext.ai.sap.com/environment", "value": "production" }
  ],
  "createdAt": "2024-03-10T09:00:00.000Z",
  "updatedAt": "2024-03-10T09:00:00.000Z"
}
```

Successful responses return code **200** and item details.

Retrieve Named Subject Patterns

To include named subject patterns in the response, set the `namedSp` query parameter to `true`.

```
curl -X GET "$AI_API_URL/v2/tcr/dataDestinations/hdl-production?namedSp=true" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP"
```

The response contains the `subjectPatterns` array and the `namedSubjectPatterns` object.

```
{
  "name": "hdl-production",
  "type": "HDL",
  "status": "ACTIVE",
  "subjectPatterns": [
    "sap.ai.core:resourceGroup:my-resource-group:dataDestination:hdl-production"
  ],
  "namedSubjectPatterns": {
    "hanaSubjectPattern": "sap.ai.core:resourceGroup:my-resource-
group:dataDestination:hdl-production",
    "aciorePklSubjectPattern": "sap.ai.core:resourceGroup:my-resource-
group:dataDestination:hdl-production"
  },
  "createdAt": "2024-03-10T09:00:00.000Z",
  "updatedAt": "2024-03-10T09:00:00.000Z"
}
```

Update a Data Destinations

You can update the `labels`, `description`, and or `config` (credentials) fields. The `name` and `type` fields cannot be changed after creation.

The patchable credential field per type:

Type	Patchable credential field(s)
S3	`access_key_id`, `secret_access_key`
GCS	`base64_encoded_private_key_data`
AZURE	`sas_token`

Type	Patchable credential field(s)
HDL	Not patchable

Send a PATCH request to update metadata labels for a data destination:

```
curl -X PATCH "$AI_API_URL/v2/tcr/dataDestinations/s3-training-data" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP" \
  -H "Content-Type: application/json" \
  -d '{
    "description": "Updated S3 bucket for ML training datasets",
    "labels": [
      { "key": "ext.ai.sap.com/environment", "value": "staging" }
    ],
    "config": {
      "access_key_id": "AKIANEWKEYEXAMPLE123",
      "secret_access_key": "newSecretKeyValue/Example+Key"
    }
  }'
```

Successful responses return code **204**.

Error Responses

- **400:** Invalid payload or label format violation
- **404:** Data destination not found

Delete Data Destinations

Send a DELETE request to the following endpoint:

```
curl -X DELETE "$AI_API_URL/v2/tcr/dataDestinations/s3-training-data" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP"
```

Successful responses return code **204**.

Error Responses

- **409:** Data destination has active dependencies that block deletion
- **404:** Data destination not found

Search Data Destinations

To filter destinations by metadata labels, use the search API:

```
curl -X POST "$AI_API_URL/v2/tcr/dataDestinations/search" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP" \
  -H "Content-Type: application/json" \
  -d '{
```

```
    "labels": [
      { "key": "ext.ai.sap.com/environment", "value": "production" }
    ]
  }'
```

Successful responses return code **200** and item details.

Entries are returned if they match all specified labels (AND logic).

```
{
  "count": 1,
  "resources": [
    {
      "name": "hdl-production",
      "type": "HDL",
      "description": "Production HANA Data Lake file container",
      "adapterType": "File",
      "labels": [
        { "key": "ext.ai.sap.com/environment", "value": "production" }
      ],
      "createdAt": "2024-03-10T09:00:00.000Z",
      "updatedAt": "2024-03-10T09:00:00.000Z"
    }
  ]
}
```

Validate a New Data Destination

You can test provider connectivity without persisting anything, to verify credentials and access before creation.

```
curl -X POST "$AI_API_URL/v2/tcr/dataDestinations/validate" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP" \
  -H "Content-Type: application/json" \
  -d '{
    "type": "S3",
    "config": {
      "bucket": "my-ml-training-bucket",
      "region": "eu-central-1",
      "access_key_id": "AKIAIOSFODNN7EXAMPLE",
      "secret_access_key": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
    }
  }'
```

Successful responses return code **200** and `status`: `OK`.

Unsuccessful responses return code **200** and `status`: `FAILED`, as well as a reason.

Validate an Existing Data Destination

You can retest provider connectivity, for example after patching your credentials.

```
curl -X POST "$AI_API_URL/v2/tcr/dataDestinations/s3-training-data/validate" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP"
```

Successful responses return code **200** and `status`: `OK`.

Error Responses return code **404**: Data destination not found

1.4.4.1.2 Tabular Artifact

A tabular artifact represents a specific file on a Data Destination that has been onboarded into the context registry.

Onboarding a tabular artifact consists of the following:

1. Reads the Core Schema Notation (CSN) schema definition (inline, by file reference, or auto-derived).
2. Creates a **virtual table** in SAP HANA pointing to the file.
3. Makes the data queryable at runtime for context selection.

CSN Schema Definition Types

You register a tabular artifact by using one of the following methods:

`definitionType`	When to use
`DOCUMENT`	Embed the CSN schema inline in the request
`REFERENCE`	CSN schema file already lives on the data destination
`AUTO`	Let the system derive the schema from the file

Artifact Status Lifecycle

The artifact lifecycle consists of the following stages:

Status	Description
`PROCESSING`	Asynchronous creation in progress
`ACTIVE`	Fully available — safe to reference in a Scenario Configuration
`ERROR`	Creation failed — check the artifact details
`DELETING`	Soft-deleted, awaiting async cleanup

Create a Tabular Artifact

> **Note**
>
> Tabular artifact creation is asynchronous. The request is accepted immediately and returns `202 Accepted`, while schema validation and resource creation run in a separate task.
>
> Before you use the artifact in a scenario configuration, poll `GET /tabularArtifacts/{name}` until the artifact status is `ACTIVE`.
>
> If creation fails, the artifact status is `ERROR` and the response includes an `errorMessage` field. To retry the operation, send another `PUT` request with the same artifact name.
>
> Specify the artifact name in the request path. Do not include the name in the request body.

Endpoint

Method	Path	Description
PUT	`/tabularArtifacts/{name}`	Create a new tabular artifact asynchronously (returns 202)

Examples

DOCUMENT Method

Use the DOCUMENT method to embed the CSN JSON directly in the request.

```
curl -X PUT "$AI_API_URL/v2/tcr/tabularArtifacts/customer-data-ta" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP" \
  -H "Content-Type: application/json" \
  -d '{
    "dataDestinationName": "s3-training-data",
    "type": "CSV",
    "path": "/data/customers/customers_2024.csv",
    "csnMetadata": {
      "entityName": "CustomerEntity",
      "selectedColumns": ["customer_id", "name", "purchase_amount", "region"],
      "definition": {
        "definitionType": "DOCUMENT",
        "document": {
          "namespace": "com.example",
          "definitions": {
            "CustomerEntity": {
              "kind": "entity",
              "elements": {
                "customer_id":      { "type": "cds.String",  "length": 50 },
                "name":             { "type": "cds.String",  "length": 200 },
                "purchase_amount":  { "type": "cds.Decimal", "precision": 15,
"scale": 2 },
                "region":           { "type": "cds.String",  "length": 50 }
              }
            }
          }
        }
      }
```

```
    }
}'
```

Output Code

```
Response — 202 Accepted:
{
  "name": "customer-data-ta"
}
```

REFERENCE Method

Use the REFERENCE method to reference a CSN file stored in the data source.

```
curl -X PUT "$AI_API_URL/v2/tcr/tabularArtifacts/orders-parquet-ta" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP" \
  -H "Content-Type: application/json" \
  -d '{
    "dataDestinationName": "s3-training-data",
    "type": "PARQUET",
    "path": "/data/orders/orders_2024.parquet",
    "csnMetadata": {
      "entityName": "OrderEntity",
      "definition": {
        "definitionType": "REFERENCE",
        "documentReference": {
          "path": "/schemas/orders_csn.json"
        }
      }
    }
  }'
```

Output Code

```
Response — 202 Accepted:
{
  "name": "orders-parquet-ta"
}
```

AUTO Method

Use the AUTO method to automatically discover schema metadata from the source.

```
curl -X PUT "$AI_API_URL/v2/tcr/tabularArtifacts/sales-data-ta" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP" \
  -H "Content-Type: application/json" \
  -d '{
    "dataDestinationName": "hdl-production",
    "type": "CSV",
    "path": "/data/sales/sales_q1.csv",
    "csnMetadata": {
      "definition": {
        "definitionType": "AUTO",
        "autoConfig": {
          "csvOptions": {
            "columnListInFirstRow": true,
            "delimiter": ","
          }
        }
```

```
}'
```

Output Code

```
Response — 202 Accepted:
{
  "name": "sales-data-ta"
}
```

Error Responses

The results may include the following error responses:

Status	When
400	Invalid payload, `path` does not start with `/`, or unsupported `type`
409	A tabular artifact with this name already exists

Next Step

For more information about tabular artifact status, see Retrieve a tabular artifact.

1.4.4.1.2.1 Manage Tabular Artifacts

Manage tabular artifacts for use in scenario configurations.

Endpoints

Method	Path	Description
GET	`/tabularArtifacts`	List all tabular artifacts
GET	`/tabularArtifacts/{name}`	Get details and full schema metadata
DELETE	`/tabularArtifacts/{name}`	Delete (async soft-delete, returns 202)
GET	`/tabularArtifacts/{name}/data`	Preview first 10 rows from the virtual table

List Tabular Artifacts

To list tabular artifacts, send a GET request to the endpoint:

```
curl -X GET "$AI_API_URL/v2/tcr/tabularArtifacts?\$top=20&\$count=true" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP"
```

Successful responses return code **200** and item details.


```
{
  "count": 2,
  "resources": [
    {
      "id": "ta-uuid-1234-5678-abcd-ef12",
      "name": "customer-data-ta",
      "tenantId": "tenant-abc123",
      "resourceGroupId": "my-resource-group",
      "dataDestinationName": "s3-training-data",
      "virtualTableName": "VT_CUSTOMER_DATA_TA_A1B2C3",
      "remoteSourceName": "RS_S3_TRAINING_DATA",
      "path": "/data/customers/customers_2024.csv",
      "type": "CSV",
      "status": "ACTIVE",
      "createdAt": "2024-03-12T08:00:00.000Z",
      "updatedAt": "2024-03-12T08:02:30.000Z"
    },
    {
      "id": "ta-uuid-9876-5432-efgh-ab34",
      "name": "sales-data-ta",
      "tenantId": "tenant-abc123",
      "resourceGroupId": "my-resource-group",
      "dataDestinationName": "hdl-production",
      "virtualTableName": null,
      "remoteSourceName": null,
      "path": "/data/sales/sales_q1.csv",
      "type": "CSV",
      "status": "PROCESSING",
      "createdAt": "2024-03-12T10:00:00.000Z",
      "updatedAt": "2024-03-12T10:00:00.000Z"
    }
  ]
}
```


Retrieve a Tabular Artifact

To retrieve a tabular artifact, send a GET request to the endpoint:

```
curl -X GET "$AI_API_URL/v2/tcr/tabularArtifacts/customer-data-ta" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP"
```

Successful responses return code **200** and item details.

The response includes `status` details. If your `status` is `ACTIVE`, you'll receive the response below:


```
{
  "id": "ta-uuid-1234-5678-abcd-ef12",
  "name": "customer-data-ta",
```

```
  "tenantId": "tenant-abc123",
  "resourceGroupId": "my-resource-group",
  "dataDestinationName": "s3-training-data",
  "virtualTableName": "VT_CUSTOMER_DATA_TA_A1B2C3",
  "remoteSourceName": "RS_S3_TRAINING_DATA",
  "path": "/data/customers/customers_2024.csv",
  "type": "CSV",
  "status": "ACTIVE",
  "csnMetadata": {
    "entityName": "CustomerEntity",
    "selectedColumns": ["customer_id", "name", "purchase_amount", "region"],
    "definition": {
      "definitionType": "DOCUMENT",
      "document": { "...": "..." }
    }
  },
  "metadata": [
    {
      "type": "csn",
      "columns": [
        { "name": "customer_id",     "cdsType": "cds.String",  "length": 50 },
        { "name": "name",            "cdsType": "cds.String",  "length": 200 },
        { "name": "purchase_amount", "cdsType": "cds.Decimal", "precision": 15,
"scale": 2 },
        { "name": "region",          "cdsType": "cds.String",  "length": 50 }
      ]
    }
  ],
  "createdAt": "2024-03-12T08:00:00.000Z",
  "updatedAt": "2024-03-12T08:02:30.000Z"
}
```

If your `status` is `PROCESSING`, you'll receive the response below:

```
{
  "id": "ta-uuid-9876-5432-efgh-ab34",
  "name": "sales-data-ta",
  "tenantId": "tenant-abc123",
  "resourceGroupId": "my-resource-group",
  "dataDestinationName": "hdl-production",
  "virtualTableName": null,
  "remoteSourceName": null,
  "path": "/data/sales/sales_q1.csv",
  "type": "CSV",
  "status": "PROCESSING",
  "csnMetadata": null,
  "metadata": null,
  "createdAt": "2024-03-12T10:00:00.000Z",
  "updatedAt": "2024-03-12T10:00:00.000Z"
}
```

Error responses return code **404**: Artifact not found.

Delete a Tabular Artifact

This endpoint soft-deletes the artifact. The virtual table and remote source are dropped asynchronously.

```
curl -X DELETE "$AI_API_URL/v2/tcr/tabularArtifacts/sales-data-ta" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP"
```

Successful responses return code **202**.

Error Responses

- **409:** Artifact is currently referenced by an active scenario configuration
- **404:** Artifact not found

Preview Data from a Tabular AI Artifact

Returns the first 10 rows from the virtual table. The artifact must have `status`: `ACTIVE`.

```
curl -X GET "$AI_API_URL/v2/tcr/tabularArtifacts/customer-data-ta/data" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP"
```

Successful responses return code **200** and item details.

```
{
  "name": "customer-data-ta",
  "columns": ["customer_id", "name", "purchase_amount", "region"],
  "data": [
    {
      "customer_id": "C-001",
      "name": "Acme Corporation",
      "purchase_amount": 15200.50,
      "region": "EMEA"
    },
    {
      "customer_id": "C-002",
      "name": "Globex Industries",
      "purchase_amount": 8750.00,
      "region": "NA"
    }
  ]
}
```

1.4.4.1.3 Scenario Configuration

A scenario configuration binds one or more tabular artifacts to a context selection strategy. This is the identifier your application passes to the Tabular AI Orchestration inference endpoint.

The following context selection strategies are available:

Strategy	Description	When to use
`embedding`	Selects rows semantically similar to the query using vector embeddings	Production — when accuracy matters
`random`	Selects rows at random	Baseline testing and development

Scenario Configuration Status Lifecycle

Status	Description
`PROCESSING`	Creation is running in the background
`ACTIVE`	Fully created and available for use
`ERROR`	Creation failed; inspect `errorMessage` and retry with `PUT` if appropriate
`DELETING`	Marked for deletion and awaiting asynchronous cleanup

Create Scenario Configuration

> **Note**
>
> Scenario configuration creation is asynchronous. The request is accepted immediately and returns `202 Accepted` while validation and background processing continue.
>
> Poll `GET /scenarioConfigurations/{name}` until the scenario configuration reaches the `ACTIVE` status.
>
> If creation fails, the scenario configuration status is set to `ERROR`, and an `errorMessage` is returned. To retry the operation, send another `PUT` request with the same scenario configuration name.
>
> Specify the scenario configuration name in the request path. Do not include it in the request body.

Endpoint

Method	Path	Description
PUT	`/scenarioConfigurations/{name}`	Create a new scenario configuration asynchronously (returns 202)

Create A Scenario Configuration with a Single Artifact

```
curl -X PUT "$AI_API_URL/v2/tcr/scenarioConfigurations/customer-churn-scenario" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP" \
  -H "Content-Type: application/json" \
  -d '{
     "description": "Customer churn prediction using embedding-based context
selection",
    "contextSelectionStrategy": "embedding",
    "tabularArtifacts": [
      { "name": "customer-data-ta" }
    ],
    "labels": [
      { "key": "ext.ai.sap.com/environment", "value": "production" },
      { "key": "ext.ai.sap.com/usecase", "value": "churn-prediction" }
    ]
  }'
```

Output Code

```
Response — 202 Accepted:
{
  "name": "customer-churn-scenario"
}
```

Create A Scenario Configuration with Multiple artifacts

A scenario can reference multiple Tabular Artifacts if the use case spans more than one table or file.

```
curl -X PUT "$AI_API_URL/v2/tcr/scenarioConfigurations/sales-forecast-scenario" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP" \
  -H "Content-Type: application/json" \
  -d '{
   "contextSelectionStrategy": "random",
    "tabularArtifacts": [
      { "name": "sales-data-ta" },
      { "name": "orders-parquet-ta" }
    ]
  }'
```

Output Code

```
Response — 202 Accepted:
{
  "name": "sales-forecast-scenario"
}
```

Next Step

For more information about scenario configuration status, see Retrieve a Scenario Configuartion.

1.4.4.1.3.1 Manage Scenario Configurations

Manage scenario configurations that define the settings and parameters required for AI scenarios.

Endpoints

Method	Path	Description
GET	`/scenarioConfigurations`	List all scenario configurations
GET	`/scenarioConfigurations/{name}`	Get details of a specific configuration

Method	Path	Description
PATCH	`/scenarioConfigurations/{name}`	Update strategy, artifacts, labels, or description
DELETE	`/scenarioConfigurations/{name}`	Delete the configuration
POST	`/scenarioConfigurations/search`	Search by label key-value pairs

List Scenario Configurations

To list all scenario configurations in your resource group, send a GET request to the following endpoint:

```
curl GET "$AI_API_URL/v2/tcr/scenarioConfigurations"?\$top=10&\$count=true" \
  --header "AI-Resource-Group: $RESOURCE_GROUP_ID" \
  --header "Authorization: Bearer $TOKEN"
```

Successful responses return code **200** and item details.

```
{
  "count": 2,
  "resources": [
    {
      "name": "customer-churn-scenario",
      "description": "Customer churn prediction using embedding-based context
selection",
      "contextSelectionStrategy": "embedding",
      "tabularArtifacts": [
        { "name": "customer-data-ta" }
      ],
      "labels": [
        { "key": "ext.ai.sap.com/environment", "value": "production" }
      ],
      "status": "ACTIVE",
      "createdAt": "2024-03-13T10:00:00.000Z",
      "updatedAt": "2024-03-13T10:00:00.000Z"
    },
    {
      "name": "sales-forecast-scenario",
      "description": null,
      "contextSelectionStrategy": "random",
      "tabularArtifacts": [
        { "name": "sales-data-ta" },
        { "name": "orders-parquet-ta" }
      ],
      "labels": [],
      "status": "ACTIVE",
      "createdAt": "2024-03-13T11:00:00.000Z",
      "updatedAt": "2024-03-13T11:00:00.000Z"
    }
  ]
}
```

Retrieve a Scenario Configuration

To retrieve a single scenario configuration, send a GET request to the following endpoint:

```
curl -X GET "$AI_API_URL/v2/tcr/scenarioConfigurations/customer-churn-scenario" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP"
```

Successful responses return code **200** and item details.

```
{
  "name": "customer-churn-scenario",
  "description": "Customer churn prediction using embedding-based context
selection",
  "contextSelectionStrategy": "embedding",
  "tabularArtifacts": [
    { "name": "customer-data-ta" }
  ],
  "labels": [
    { "key": "ext.ai.sap.com/environment", "value": "production" },
    { "key": "ext.ai.sap.com/usecase", "value": "churn-prediction" }
  ],
  "status": "ACTIVE",
  "createdAt": "2024-03-13T10:00:00.000Z",
  "updatedAt": "2024-03-13T10:00:00.000Z"
}
```

Update a Scenario Configuration

You can update any combination of the following: `contextSelectionStrategy`, `tabularArtifacts`, `labels`, or `description`. The `name` cannot be changed. At least one field must be present.

Update Strategy

```
curl -X PATCH "$AI_API_URL/v2/tcr/scenarioConfigurations/customer-churn-
scenario" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP" \
  -H "Content-Type: application/json" \
  -d '{
    "contextSelectionStrategy": "random"
  }'
```

Replace Tabular Artifacts

```
curl -X PATCH "$AI_API_URL/v2/tcr/scenarioConfigurations/customer-churn-
scenario" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP" \
  -H "Content-Type: application/json" \
  -d '{
    "tabularArtifacts": [
      { "name": "customer-data-ta" },
      { "name": "orders-parquet-ta" }
    ]
  }'
```

Successful responses return code **204**.

Error Responses

- **400:** Invalid payload or label format violation
- **404:** Scenario configuration not found

Delete a Scenario Configuration

Send a DELETE request to the endpoint:

```
curl -X DELETE "$AI_API_URL/v2/tcr/scenarioConfigurations/customer-churn-
scenario" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP"
```

Successful responses return code **204**.

Error responses return code **404**: Scenario configuration not found

Search for a Configuration by Labels

Returns scenario configurations matching all specified labels (AND logic).

```
curl -X POST "$AI_API_URL/v2/tcr/scenarioConfigurations/search" \
  -H "Authorization: Bearer $TOKEN" \
  -H "AI-Resource-Group: $RESOURCE_GROUP" \
  -H "Content-Type: application/json" \
  -d '{
    "labels": [
      { "key": "ext.ai.sap.com/environment", "value": "production" }
    ]
  }'
```

Successful responses return code **200** and item details.

```
{
  "count": 1,
  "resources": [
    {
      "name": "customer-churn-scenario",
      "description": "Customer churn prediction using embedding-based context
selection",
      "contextSelectionStrategy": "embedding",
      "tabularArtifacts": [
        { "name": "customer-data-ta" }
      ],
      "labels": [
        { "key": "ext.ai.sap.com/environment", "value": "production" }
      ],
      "createdAt": "2024-03-13T10:00:00.000Z",
      "updatedAt": "2024-03-13T10:00:00.000Z"
    }
  ]
}
```

Successful responses with no matches return code **200** and empty item details.

```
{
  "count": 0,
  "resources": []
}
```

1.4.4.1.4 Cleanup (Optional)

You can remove design-time resources independently.

Resource	Endpoint	Behavior
Scenario Configuration	`DELETE /v2/tcr/scenarioConfigurations/{name}`	Removes the scenario only
Tabular Artifact	`DELETE /v2/tcr/tabularArtifacts/{name}`	Removes the artifact and its virtual table
Data Destination	`DELETE /v2/tcr/dataDestinations/{name}`	Removes destination

1.4.4.2 Runtime Setup

Overview

The runtime setup for tabular orchestration in SAP AI Core consists of creating a deployment one time and then consuming the deployed service at runtime.

Step 1: Create a Deployment for Tabular Orchestration

Create the deployment for tabular orchestration in SAP AI Core. This setup step is required only once per resource group.

For the complete setup procedure, see **Create a Tabular AI Deployment [page 629]**.

Step 2: Consume Tabular AI Orchestration Service

After the deployment is in the `RUNNING` state, consume the tabular AI orchestration service using the deployment URL.

For detailed consumption steps and API usage, see **Consume Tabular Orchestration [page 631]**.

1.4.4.2.1 Create a Tabular AI Deployment

Create a configuration and deployment in SAP AI Core to access tabular orchestration.

Prerequisites

- You have an SAP AI Core service instance and service key. For more information, see SAP AI Core Initial Setup Documentation.
- You're using the `extended` service plan. For more information, see Service Plans.
- You have created a resource group. For more information, seeCreate a Resource Group for Grounding [page 397].
- Make sure that you've set the following **headers**:

Key	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps

Make sure you have set the following variable:

- **$AI_API_URL**: The base URL of your SAP AI Core environment. You can set this URL as an environment variable.

> **Note**
>
> Use the same resource group for all generative AI activities. To use a different resource group, repeat these steps for each resource group.

Access Tabular Orchestration

- Create a configuration
- Create a deployment
- When the deployment reaches the `RUNNING` state, retrieve the `deploymentUrl`.
- Use the `deploymentUrl` to access tabular orchestration.

Check Scenario and Executable

Check Scenario

Check that you have access to the `tabular-orchestration` scenario:

```
curl GET "$AI_API_URL/v2/lm/scenarios/tabular-orchestration" \
  --header "AI-Resource-Group: $RESOURCE_GROUP_ID" \
  --header "Authorization: Bearer $TOKEN"
```

Result

You receive a successful response including the scenario ID `tabular-orchestration`.

Check Executable

Check that you have access to the `tabular-orchestration` executable:

```
curl GET "$AI_API_URL/v2/lm/scenarios/tabular-orchestration/executables/tabular-
orchestration" \
  --header "AI-Resource-Group: $RESOURCE_GROUP_ID" \
  --header "Authorization: Bearer $TOKEN"
```

Result

You receive a successful response including the executable ID `tabular-orchestration`.

Create a Configuration

Send a POST request to create a configuration:

```
curl POST "$AI_API_URL/v2/lm/configurations" \
  --header "AI-Resource-Group: $RESOURCE_GROUP_ID" \
  --header "Authorization: Bearer $TOKEN" \
  --header "Content-Type: application/json" \
  --data '{
    "name": "<yourNameChoice>",
    "executableId": "tabular-orchestration",
    "scenarioId": "tabular-orchestration"
  }'
```

Result

You receive a unique `configurationId` in the response. Set the variable **`CONFIGURATION_ID`** to this value.

Create a Deployment

Send a POST request to create a deployment:

```
curl POST "$AI_API_URL/v2/lm/deployments" \
  --header "AI-Resource-Group: $RESOURCE_GROUP_ID" \
  --header "Authorization: Bearer $TOKEN" \
  --header "Content-Type: application/json" \
```

```
  --data '{
    "configurationId": "<yourConfigurationId>"
  }'
```

Result

You receive a unique `deploymentId` in the response. Set the variable **`DEPLOYMENT_ID`** to this value.

Next Steps

Retrieve the details of your deployment by sending a GET request:

```
curl GET "$AI_API_URL/v2/lm/deployments/{deploymentId}"
```

When the deployment reaches the `RUNNING` state, note the `deploymentUrl` in the response.

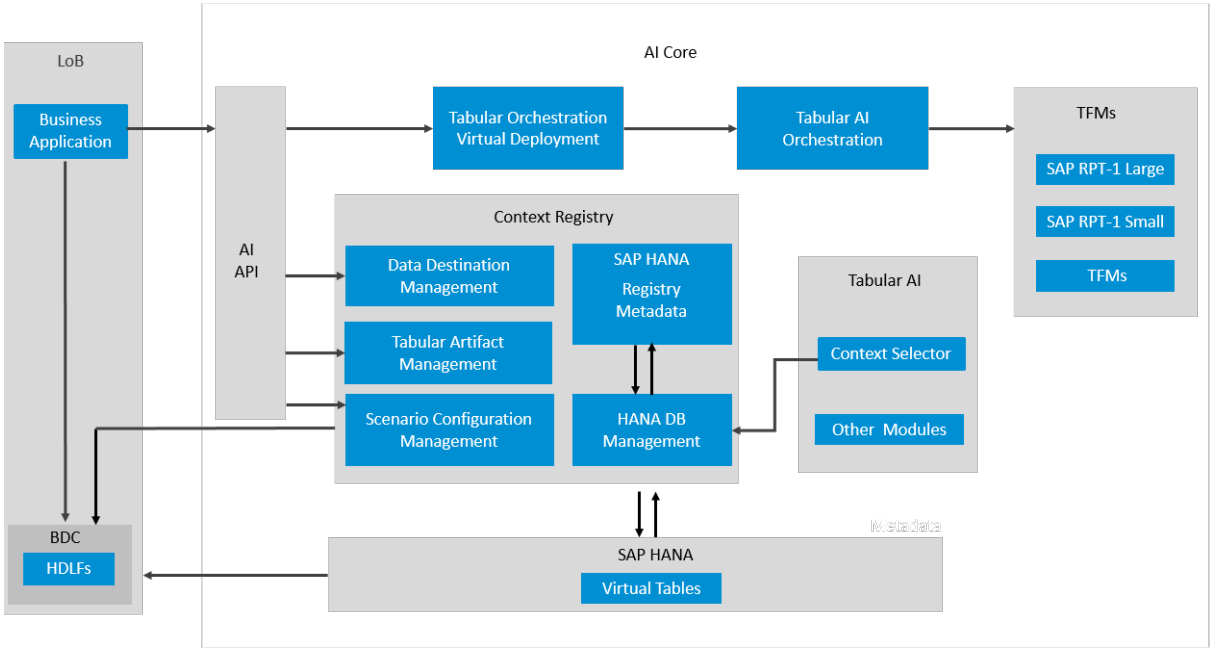
Use the `deploymentUrl` to access tabular orchestration.

1.4.4.2.2 Consume Tabular Orchestration

The Tabular AI Orchestration service manages the end-to-end prediction workflow, from context retrieval to model inference.

The Tabular AI Orchestration service is the runtime inference layer of the Tabular AI platform. It provides a unified prediction endpoint that orchestrates the end-to-end prediction workflow. The service validates incoming requests, retrieves context rows from the Context Selector service, combines them with any user-provided context, and forwards the assembled payload to a Tabular Foundation Model (TFM) for inference.

> **Note**
>
> Before calling the orchestration endpoint, complete the design-time setup. A data destination, tabular artifact, and scenario configuration must exist in the Context Registry.



For more information, see the following topics:

- API Specification [page 632]
- Troubleshooting [page 650]
- Prediction Examples for Tabular AI Orchestration [page 642]

1.4.4.2.2.1 API Specification

Use the Tabular AI Orchestration Prediction API to submit inference requests, configure context selection strategies, and generate classification or regression predictions with supported tabular foundation models.

API Specification

Tabular AI Orchestration exposes a single prediction endpoint for inference requests.

```
curl POST "$DEPLOYMENT_URL/predict"
```

Required Headers

Header	Type	Description
`ai-resource-group`	string	Resource group identifier for SAP AI Core
`Content-Type`	string	Must be `application/json`

Request Parameters

Parameter	Type	Required	Default	Description
`modelName`	string	Yes	-	Name of the tabular foundation model to use for predictions. For more information, see Supported Models .
`scenarioConfigName`	string	Yes	-	Name of the scenario configuration created during design-time setup. This configuration links to the tabular artifact registered in the context registry.
`predictionConfig`	object	Yes	-	Configuration that controls how predictions are generated by the selected tabular foundation model.
`predictionConfig.targetColumns`	array	Yes	-	List of target column configurations. Specify at least one target column.
`predictionConfig.targetColumns[].name`	string	Yes	-	Name of the column to predict. The column must exist in the query data.
`predictionConfig.targetColumns[].predictionPlaceholder`	string	No	[PREDICT]	Placeholder value that identifies cells to predict. Default: [PREDICT].
`predictionConfig.targetColumns[].taskType`	string	No	-	Type of prediction task. If you don't specify this value, the model determines the task type automatically. Supported values are `classification` and `regression`. For more information, see Prediction Task Types .
`predictionConfig.targetColumns[].topK`	integer	No	1	Number of labels to return for each prediction. Applies only to classification tasks. Must be greater than or equal to 1. Default: 1.
`modelConfig.*`	Any	No	-	Additional configuration parameters that are specific to the tabular foundation model and that are passed to the selected model.
`contextSelectionConfig`	object	No	-	Configuration for context selection. If you don't specify this value, the service uses the default settings.

Parameter	Type	Required	Default	Description
`contextSelectionConfig.numRows`	integer	No	-	Number of context rows to retrieve from the registered tabular artifact through the context selector service. If this value is `0` or `null`, context selection is skipped. The value must be greater than or equal to 0.
`contextSelectionConfig.indexColumn`	String	No	-	Index column used for context selection. Required when the `heuristic` strategy is selected.
`contextSelectionConfig.strategy`	string	No	random	Context selection strategy. Supported values are `none`, `random`, `heuristic`, and `auto`. The default is `random`.
`contextSelectionConfig.strategyConfig`	object	No	-	Strategy-specific configuration parameters. For more information, see Context Selection Strategy Configurations.
`columns`	object	Yes*	-	Query rows in **columnar** format. Maps column names to arrays of values. All arrays must have the same length.
`contextColumns`	object	No	null	User-provided context rows in **columnar** format. Applicable only when the `columns` parameter is provided. Must use the same schema as `columns`.
`rows`	array	Yes*	-	Query rows in **row** format. Each object represents a table row. All objects must use the same keys.
`contextRows`	array	No	null	User-provided context rows in **row** format. Applicable only when the `rows` parameter is provided. Must use the same schema as `rows`.

Data Orientation

Provide query data in one format only, either `columns` or `rows`. Do not provide both formats in the same request. Provide context data in the same format as the query data. For example, if you use `columns`, provide context data in `contextColumns`, not in `contextRows`.

Prediction Configurations

For information about the parameters available for each tabular foundation model, see the corresponding model documentation.

Input Validation

The API validates requests at two levels: at schema level and at the level of the business logic.

Schema Validation

- Data format: Provide exactly one query data format — either `columns` or `rows`.
- Context format: Use `contextColumns` with columnar query data and `contextRows` with row-based query data.

- Column consistency: In columnar format, all column arrays must contain the same number of values.
- Row consistency: In row format, all row objects must contain the same set of keys.
- Schema matching: Context data must use the same column names as the query data.
- Minimum data: Provide at least one row or one column. Each column must contain at least one value.
- Target columns: Specify at least one target column in `predictionConfig.targetColumns`.
- `topK`: If specified, the value must be 1 or greater. This parameter is supported only for classification tasks.
- `numRows`: If specified, the value must be 0 or greater.

Business Logic Validation

- Model limits: The number of query rows must not exceed the model's maximum supported value. See #supported_models.
- Model name: Use a name of a support tabular foundation model.
- Strategy: Specify one of the supported strategies: `none`, `random`, `heuristic`, or `auto`.

> **Note**
>
> If validation fails, the API returns a `400 Bad Request` response with detailed error information.

Response Headers

Header	Description
`X-Response-Time`	Total processing time
`X-Correlation-ID`	Correlation identifier used for tracing
`X-Context-Selection-Time`	Context-selection execution time
`X-TFM-Inference-Time`	Model-inference execution time

> **Note**
>
> Use these headers to monitor requests, troubleshoot issues, and analyze performance.

Response Schema

The Tabular Orchestration service returns the response from the tabular foundation model without modification.

The `SAP-RPT-1` model returns the following response fields:

Field	Type	Description
`id`	string	Prediction request identifier

Field	Type	Description
`metadata`	object	Prediction metadata
`metadata.num_columns`	integer	Total number of columns
`metadata.num_rows`	integer	Total number of rows
`metadata.num_predictions`	integer	Total predictions returned
`metadata.num_query_rows`	integer	Total query rows
`predictions`	array	Prediction results
`status`	object	Status object
`status.code`	integer	Status code
`status.message`	string	Status message
`additional_information`	array	Warnings and informational messages

Each object in the `predictions` array can contain the following information:

- The value of the configured `indexColumn`, if specified in the request.
- For each target column, an array of prediction objects that includes:
  - `prediction`
  - `confidence` (classification predictions only)
  - `confidence_interval`

Response Status Codes

Status Code	Error Code	Description
200	-	Prediction completed successfully
400	`VALIDATION_ERROR`	Request validation failed
401	-	Authentication failed
404	`MODEL_NOT_FOUND`	Model not available
404	`SCENARIO_CONFIG_NOT_FOUND_ERROR`	Scenario configuration not found
500	`CONTEXT_SELECTOR_ERROR`	Context selector failure
500	`SAP_RPT1_SMALL_ERROR`	Failure with `sap-rpt-1-small`
500	`SAP_RPT1_LARGE_ERROR`	Failure with `sap-rpt-1-large`
502	`BAD_GATEWAY`	External service failure
504	`GATEWAY_TIMEOUT`	Request timeout

Error responses follow a standard structure:

```
{
  "error": {
```

```
    "code": "VALIDATION_ERROR",
    "message": "Query rows (150) exceeds maximum (100)"
  }
}
```

Supported Models

To select the model, specify it using the `ModelName`.

Model	Model Name	Maximum Context Rows	Maximum Query Rows
SAP-RPT-1 Small	`sap-rpt-1-small`	2048	128
SAP-RPT-1 Large	`sap-rpt-1-large`	8192	128
SAP-RPT-1.5	`sap-rpt-1.5`	2048	128
SAP-RPT-1.5 Large	`sap-rpt-1.5-large`	8192	128

Context Data Formats

The service supports columnar and row-wise data formats.

Columnar Format

Columnar format represents data as an object where each key is a column name and each value is an array of values.

```
{
  "id": [1, 2, 3],
  "category": ["[PREDICT]", "[PREDICT]", "[PREDICT]"],
  "price": [99.99, 49.99, 29.99]
}
```

Row-Wise Format

Row-wise format represents data as an array of objects where each object corresponds to a row.

```
[
  { "id": 1, "category": "[PREDICT]", "price": 99.99 },
  { "id": 2, "category": "[PREDICT]", "price": 49.99 },
  { "id": 3, "category": "[PREDICT]", "price": 29.99 }
]
```

Context Selection Strategies

The `contextSelectionConfig.strategy` parameter controls how context rows are selected from a registered tabular artifact.

Supported values are:

- `random`
- `heuristic`
- `none`
- `auto`

Random Strategy

The random strategy randomly samples `numRows` from the tabular artifact.

This strategy is used by default.

Execution time remains relatively constant because performance does not significantly depend on the number of query rows or the number of columns.

Use Case: Use this strategy when you need fast and predictable response times.

Random strategy parameters

Field	Type	Required	Default	Description
`contextSelectionConfig.strategyConfig.deterministic`	boolean	No	-	Uses a Linear Congruential Generator seeded by the index column and produces the same row ordering for identical data.
`contextSelectionConfig.strategyConfig.indexColumn`	string	No	-	Name of the unique row identifier column. Required when deterministic mode is enabled.

Random strategy example

```
{
  "contextSelectionConfig": {
    "numRows": 200,
    "strategy": "random",
    "strategyConfig": {
      "deterministic": true,
      "indexColumn": "id"
    }
  }
}
```

Heuristic Strategy

The heuristic strategy selects context rows by scoring similarity between query rows and artifact rows.

The service automatically detects numeric, string, and date columns and applies the appropriate matching logic.

Execution time depends on:

- Tabular artifact size
- Number of query rows
- Number of columns
- Prediction configuration complexity

Use Case: Use this strategy to retrieve context rows that closely match the query rows based on numeric, text, or date features.

Heuristic Strategy Parameters

Field	Type	Required	Default	Description
`indexColumn`	string	Yes	-	Unique row identifier used for scoring and de-duplication.
`methodRatio`	float	No	0.67	Defines the proportion of rows selected through similarity scoring. Remaining rows are selected randomly. Enter a value between `0` and `1`.
`poolSize`	int	No	null	Limits similarity scoring to a randomly sampled subset of rows. If not specified, all rows are evaluated.
`fuzzyNumPct`	float	No	0.1	Defines the allowed deviation for numeric comparisons as a percentage of the query value. For example, `0.1` allows a deviation of ±10%.
`fuzzyDateDays`	int	No	30	Defines the allowed deviation for date comparisons in days. Set to `0` to require an exact date match.
`allowTimeoutFallback`	boolean	No	False	Enables fallback to random sampling when similarity scoring times out. This parameter is set automatically when the strategy is `AUTO`.

Heuristic Strategy Example

```
{
  "contextSelectionConfig": {
    "numRows": 200,
    "strategy": "heuristic",
    "strategyConfig": {
      "indexColumn": "id",
      "methodRatio": 0.67,
      "chunkSize": 100,
      "chunkThreshold": 150,
      "poolSize": null,
      "fuzzyDateDays": 30,
      "fuzzyNumPct": 0.1,
      "allowTimeoutFallback": true
      }
  }
}
```

No Context Selection

Use the `none` strategy to skip context selection entirely.

This configuration is useful when context is supplied directly through `contextColumns` or `contextRows`.

```
{
  "contextSelectionConfig": {
    "strategy": "none"
  }
}
```

Prediction Task Types

The orchestration service supports classification and regression prediction tasks.

Configure the task type for each target column by using `predictionConfig.targetColumns[*].taskType`.

The selected task type determines how the tabular foundation model interprets and predicts values for cells marked with a prediction placeholder.

For complete response payload examples, see the Example Payloads for Inferencing: sap-rpt-1 [page 124].

Classification Predictions

Use the classification task type when the target column contains categorical values such as product categories, job codes, or status values.

The model predicts a discrete label for each prediction placeholder.

Supported column types are string columns that contain a finite set of distinct values.

Classification Configuration Example

```
{
  "name": "jobcode",
  "predictionPlaceholder": "[PREDICT]",
  "taskType": "classification",
  "topK": 1
}
```

Classification Response Example

```
{
  "jobcode": [
    {
      "confidence": 0.53,
      "confidence_interval": null,
      "prediction": "AEng"
    }
  ],
  "requisition_id": 2927
}
```

- `prediction` contains the predicted category.
- `confidence` contains a confidence score between 0 and 1.
- `confidence_interval` contains the confidence interval when available.

Returning Multiple Predictions with topK

Set `topK` to a value greater than 1 to return multiple ranked candidate predictions for a target column.

```
{
  "name": "jobcode",
  "taskType": "classification",
  "topK": 3
}
```

Returned candidates are ordered by confidence score in descending order.

Regression Predictions

Use the regression task type when the target column contains continuous numeric values such as prices, quantities, or scores.

The model predicts a numeric value for each prediction placeholder.

Supported column types are numeric columns, including integers and decimal values.

Regression Configuration Example

```
{
  "name": "price",
  "predictionPlaceholder": "[PREDICT]",
  "taskType": "regression"
}
```

> **Note**
>
> The topK parameter is not supported for regression tasks.

Regression Response Example

```
{
  "price": [
    {
      "confidence_interval": null,
      "prediction": 89.99
    }
  ],
  "id": 1
}
```

- `prediction` contains the predicted numeric value.
- `confidence_interval` contains the confidence interval when available.

The service supports nested filter conditions that combine multiple logical expressions.

Multi-Target Prediction

The orchestration service supports predicting multiple target columns within the same request.

Each target column can define its own prediction configuration, placeholder value, and task type. See Example:3 Multi-target Prediction with context selection for a complete request and response.

```
"targetColumns": [
  { "name": "category", "taskType": "classification" },
  { "name": "price", "taskType": "regression" }
]
```

> **Note**
>
> Multi-target prediction is supported only for the SAP RPT model family.

Custom Prediction Placeholders

You can define custom placeholder values for individual target columns.

```
{
  "targetColumns": [
    {
      "name": "category",
      "predictionPlaceholder": "[MISSING]",
      "taskType": "classification"
    },
    {
      "name": "price",
      "predictionPlaceholder": "???",
      "taskType": "regression"
    }
  ]
}
```

The service replaces cells containing the configured placeholder with predicted values generated by the selected model.

1.4.4.2.2.2 Prediction Examples for Tabular AI Orchestration

The following examples show common prediction scenarios supported by the Tabular AI Orchestration service.

Example	Description
Prediction with context selection	Retrieves context rows from a registered tabular artifact before running predictions.

Example	Description
Prediction with context selection and user-provided context	Combines user-provided context rows with context rows selected by the service.
Multi-target prediction	Predicts multiple target columns in a single request.
Classification with top-K predictions	Returns multiple ranked prediction candidates for classification tasks.

Example 1: Prediction with Context Selection

```
curl -X POST "$DEPLOYMENT_URL/predict" \
  -H "Content-Type: application/json" \
  -H "ai-main-tenant: <your-tenant-uuid>" \
  -H "ai-resource-group: <your-resource-group>" \
  -d '{
    "modelName": "sap-rpt-1-small",
    "scenarioConfigName": "my_scenario_config_name",
    "contextSelectionConfig": {
      "numRows": 200,
      "strategy": "random"
    },
    "predictionConfig": {
      "indexColumn": "requisition_id",
      "targetColumns": [
        {
          "name": "jobcode",
          "predictionPlaceholder": "[PREDICT]",
          "taskType": "classification"
        }
      ]
    },
    "columns": {
      "requisition_id": ["2927", "3144", "2403"],
      "country_id": ["US", "US", "US"],
      "hiring_manager_id": ["73F840E39C51453EBCF034735E6A28F3",
"73F840E39C51453EBCF034735E6A28F3", "73F840E39C51453EBCF034735E6A28F3"],
      "recruiter_id": ["131A0E2C84E54433B25560D7087D064F",
"131A0E2C84E54433B25560D7087D064F", "131A0E2C84E54433B25560D7087D064F"],
      "jobcode": ["[PREDICT]", "[PREDICT]", "[PREDICT]"]
    }
  }'
```

Output Code

```
{
  "id": "b782effe-6f2b-46ea-84cf-6c28790cf835",
  "metadata": {
    "num_columns": 5,
    "num_rows": 203,
    "num_predictions": 3,
    "num_query_rows": 3
  },
  "predictions": [
    {
      "jobcode": [
        {
          "confidence": 0.53,
          "confidence_interval": null,
          "prediction": "AEng"
```

```
        }
      ],
      "requisition_id": 2927
    },
    {
      "jobcode": [
        {
          "confidence": 0.53,
          "confidence_interval": null,
          "prediction": "AEng"
        }
      ],
      "requisition_id": 3144
    },
    {
      "jobcode": [
        {
          "confidence": 0.53,
          "confidence_interval": null,
          "prediction": "AEng"
        }
      ],
      "requisition_id": 2403
    }
  ],
  "status": {
    "code": 0,
    "message": "ok"
  }
}
```


Example 2: Predictions with context selection and user-provided context

```
curl -X POST "$DEPLOYMENT_URL/predict" \
  -H "Content-Type: application/json" \
  -H "ai-main-tenant: <your-tenant-uuid>" \
  -H "ai-resource-group: <your-resource-group>" \
  -d '{
    "modelName": "sap-rpt-1-small",
    "scenarioConfigName": "my_scenario_config_name",
    "contextSelectionConfig": {
      "numRows": 200,
      "strategy": "random"
    },
    "predictionConfig": {
      "indexColumn": "requisition_id",
      "targetColumns": [
        {
          "name": "jobcode",
          "predictionPlaceholder": "[PREDICT]",
          "taskType": "classification"
        }
      ]
    },
    "columns": {
      "requisition_id": ["2927", "3144", "2403"],
      "country_id": ["US", "US", "US"],
      "hiring_manager_id": ["73F840E39C51453EBCF034735E6A28F3",
"73F840E39C51453EBCF034735E6A28F3", "73F840E39C51453EBCF034735E6A28F3"],
      "recruiter_id": ["131A0E2C84E54433B25560D7087D064F",
"131A0E2C84E54433B25560D7087D064F", "131A0E2C84E54433B25560D7087D064F"],
      "jobcode": ["[PREDICT]", "[PREDICT]", "[PREDICT]"]
```

```
    },
    "contextColumns": {
      "requisition_id": ["29271", "31441", "24031"],
      "country_id": ["US", "US", "US"],
      "hiring_manager_id": ["73F840E39C51453EBCF034735E6A28F3",
"73F840E39C51453EBCF034735E6A28F3", "73F840E39C51453EBCF034735E6A28F3"],
      "recruiter_id": ["131A0E2C84E54433B25560D7087D064F",
"131A0E2C84E54433B25560D7087D064F", "131A0E2C84E54433B25560D7087D064F"],
      "jobcode": ["ABcd", "ABcd", "ABcd"]
    }
  }'
```

Output Code


```
{
  "id": "3768f648-1f4e-443b-832d-e474eefa1ec4",
  "metadata": {
    "num_columns": 5,
    "num_rows": 206,
    "num_predictions": 3,
    "num_query_rows": 3
  },
  "predictions": [
    {
      "jobcode": [
        {
          "confidence": 0.5,
          "confidence_interval": null,
          "prediction": "AEng"
        }
      ],
      "requisition_id": 2927
    },
    {
      "jobcode": [
        {
          "confidence": 0.5,
          "confidence_interval": null,
          "prediction": "AEng"
        }
      ],
      "requisition_id": 3144
    },
    {
      "jobcode": [
        {
          "confidence": 0.5,
          "confidence_interval": null,
          "prediction": "AEng"
        }
      ],
      "requisition_id": 2403
    }
  ],
  "status": {
    "code": 0,
    "message": "ok"
  }
}
```

Example 3: Multi-target Prediction with context selection

```
curl -X POST "$DEPLOYMENT_URL/predict" \
  -H "Content-Type: application/json" \
  -H "ai-main-tenant: <your-tenant-uuid>" \
  -H "ai-resource-group: <your-resource-group>" \
  -d '{
    "modelName": "sap-rpt-1-small",
    "scenarioConfigName": "my_scenario_config_name",
    "contextSelectionConfig": {
      "numRows": 200,
      "strategy": "random"
    },
    "predictionConfig": {
      "indexColumn": "requisition_id",
      "targetColumns": [
        {
          "name": "jobcode",
          "predictionPlaceholder": "[PREDICT]",
          "taskType": "classification"
        },
        {
          "name": "country_id",
          "predictionPlaceholder": "[PREDICT]",
          "taskType": "classification"
        }
      ]
    },
    "columns": {
      "requisition_id": ["2927", "3144", "2403"],
      "country_id": ["[PREDICT]", "[PREDICT]", "[PREDICT]"],
      "hiring_manager_id": ["73F840E39C51453EBCF034735E6A28F3",
"73F840E39C51453EBCF034735E6A28F3", "73F840E39C51453EBCF034735E6A28F3"],
      "recruiter_id": ["131A0E2C84E54433B25560D7087D064F",
"131A0E2C84E54433B25560D7087D064F", "131A0E2C84E54433B25560D7087D064F"],
      "jobcode": ["[PREDICT]", "[PREDICT]", "[PREDICT]"]
    }
  }'
```

Output Code


```
{
  "id": "98492b16-c21d-4233-a701-56897226f884",
  "metadata": {
    "num_columns": 5,
    "num_rows": 203,
    "num_predictions": 6,
    "num_query_rows": 3
  },
  "predictions": [
    {
      "country_id": [
        {
          "confidence": 1.0,
          "confidence_interval": null,
          "prediction": "US"
        }
      ],
      "jobcode": [
        {
          "confidence": 0.53,
          "confidence_interval": null,
          "prediction": "AEng"
        }
      ],
```

```
      "requisition_id": 2927
    },
    {
      "country_id": [
        {
          "confidence": 1.0,
          "confidence_interval": null,
          "prediction": "US"
        }
      ],
      "jobcode": [
        {
          "confidence": 0.53,
          "confidence_interval": null,
          "prediction": "AEng"
        }
      ],
      "requisition_id": 3144
    },
    {
      "country_id": [
        {
          "confidence": 1.0,
          "confidence_interval": null,
          "prediction": "US"
        }
      ],
      "jobcode": [
        {
          "confidence": 0.53,
          "confidence_interval": null,
          "prediction": "AEng"
        }
      ],
      "requisition_id": 2403
    }
  ],
  "status": {
    "code": 0,
    "message": "ok"
  }
}
```


Example 4: Classification with Top-K Predictions

```
curl -X POST "$DEPLOYMENT_URL/predict" \
  -H "Content-Type: application/json" \
  -H "ai-main-tenant: <your-tenant-uuid>" \
  -H "ai-resource-group: <your-resource-group>" \
  -d '{
    "modelName": "sap-rpt-1-large",
    "scenarioConfigName": "my_scenario_config_name",
    "contextSelectionConfig": {
      "numRows": 200,
      "strategy": "random"
    },
    "predictionConfig": {
      "indexColumn": "requisition_id",
      "targetColumns": [
        {
          "name": "jobcode",
          "predictionPlaceholder": "[PREDICT]",
```

```
          "taskType": "classification",
          "topK": 4
        },
        {
          "name": "country_id",
          "predictionPlaceholder": "[PREDICT]",
          "taskType": "classification",
          "topK": 2
        }
      ]
    },
    "columns": {
      "requisition_id": ["2927", "3144", "2403"],
      "country_id": ["US", "IN", "[PREDICT]"],
      "hiring_manager_id": ["73F840E39C51453EBCF034735E6A28F3",
"73F840E39C51453EBCF034735E6A28F3", "73F840E39C51453EBCF034735E6A28F3"],
      "recruiter_id": ["131A0E2C84E54433B25560D7087D064F",
"131A0E2C84E54433B25560D7087D064F", "131A0E2C84E54433B25560D7087D064F"],
      "jobcode": ["[PREDICT]", "[PREDICT]", "[PREDICT]"]
    }
  }'
```

Output Code


```
{
  "id": "d840df26-2fc5-4823-9295-78ed614cf1bd",
  "metadata": {
    "num_columns": 5,
    "num_rows": 203,
    "num_predictions": 6,
    "num_query_rows": 3
  },
  "predictions": [
    {
      "country_id": [
        {
          "confidence": 1.0,
          "confidence_interval": null,
          "prediction": "US"
        },
        {
          "confidence": 0.0,
          "confidence_interval": null,
          "prediction": "IN"
        }
      ],
      "jobcode": [
        {
          "confidence": 0.53,
          "confidence_interval": null,
          "prediction": "AEng"
        },
        {
          "confidence": 0.45,
          "confidence_interval": null,
          "prediction": "Atech"
        },
        {
          "confidence": 0.01,
          "confidence_interval": null,
          "prediction": "ADMIN"
        },
        {
          "confidence": 0.01,
          "confidence_interval": null,
          "prediction": "00001"
        }
```

```
      ],
      "requisition_id": 2927
    },
    {
      "country_id": [
        {
          "confidence": 1.0,
          "confidence_interval": null,
          "prediction": "US"
        },
        {
          "confidence": 0.0,
          "confidence_interval": null,
          "prediction": "IN"
        }
      ],
      "jobcode": [
        {
          "confidence": 0.53,
          "confidence_interval": null,
          "prediction": "AEng"
        },
        {
          "confidence": 0.45,
          "confidence_interval": null,
          "prediction": "Atech"
        },
        {
          "confidence": 0.01,
          "confidence_interval": null,
          "prediction": "ADMIN"
        },
        {
          "confidence": 0.01,
          "confidence_interval": null,
          "prediction": "00001"
        }
      ],
      "requisition_id": 3144
    },
    {
      "country_id": [
        {
          "confidence": 1.0,
          "confidence_interval": null,
          "prediction": "US"
        },
        {
          "confidence": 0.0,
          "confidence_interval": null,
          "prediction": "IN"
        }
      ],
      "jobcode": [
        {
          "confidence": 0.53,
          "confidence_interval": null,
          "prediction": "AEng"
        },
        {
          "confidence": 0.45,
          "confidence_interval": null,
          "prediction": "Atech"
        },
        {
          "confidence": 0.01,
          "confidence_interval": null,
          "prediction": "ADMIN"
```

```
        },
        {
          "confidence": 0.01,
          "confidence_interval": null,
          "prediction": "00001"
        }
      ],
      "requisition_id": 2403
    }
  ],
  "status": {
    "code": 0,
    "message": "ok"
  }
}
```

> **Note**
>
> The response returns `203` rows in `num_rows`, including 200 rows from the orchestrated context selection configured in `contextSelectionConfig.numRows` and 3 query rows provided in the request.

1.4.4.2.2.3 Troubleshooting

Use the following troubleshooting information to identify and resolve common validation, service, and performance issues when calling the API.

Common Validation Errors

- Error: "Cannot provide both 'rows' and 'columns'"
  Resolution: Provide query data in either `columns` format or `rows` format, but not both.
- Error: "All columns must have the same number of rows"
  Resolution: Ensure that all column arrays contain the same number of values.
- Error: "Context columns schema does not match query columns schema"
  Resolution: Ensure that `contextColumns` uses the same column names as `columns`.
- Error: "Must provide either 'rows' or 'columns' for query data"
  Resolution: Include either `rows` or `columns` in the request.
- Error: "When using 'rows' for query data, use 'contextRows' for context data"
  Resolution: Use `contextRows` when the request uses `rows` for query data.

Common Service Errors

- Error: "SCENARIO_CONFIG_NOT_FOUND_ERROR"
  Resolution: Verify that the scenario configuration exists in the Context Registry and check the scenario configuration name for typographical errors.

- Error: "CONTEXT_SELECTOR_ERROR"
  Resolution: Verify that the Context Selector service is available, the referenced Tabular Artifact is accessible, and all filter conditions are valid.
- Error: "MODEL_NOT_FOUND"
  Resolution: Verify that `modelName` specifies a supported model and check the model name for typographical errors.
- Error: "SAP_RPT1_SMALL_ERROR" or "SAP_RPT1_LARGE_ERROR"
  Resolution: Verify that the input data format is valid, the target column exists in the dataset, and the request does not exceed model limits.

Performance Issues

- Issue: Requests take longer than 10 seconds.
  Resolution: Reduce the `numRows` value, decrease the number of columns in the query data, use a smaller Tabular Artifact, or use a different context selection strategy.
- Issue: Requests return a high number of errors.
  Resolution: Verify authentication headers, ensure that the Context Selector and TFM services are available, and check network connectivity to downstream services.

Performance Optimization

To optimize performance, consider the following recommendations:

- Use `strategy="none"` if you manage context data outside the orchestration workflow.
- Use `strategy="random"` for consistent and fast context selection.
- Use `strategy="heuristic"` when prediction quality is more important than context selection time.
- Use only the context data required for the prediction task. Larger context datasets can increase context selection and inference time.

Error Handling

To improve reliability, follow these recommendations:

- Implement exponential backoff when retrying requests that return 5xx errors.
- Do not retry requests that return 4xx errors. These errors typically indicate an issue with the request.
- Use the `X-Correlation-ID` header to troubleshoot failed requests.
- Validate data schemas before sending requests.
- Ensure that all columns contain the same number of values in columnar format.
- Ensure that all row objects contain the same set of keys in row format.
- Verify that all target columns exist in the query data.
- Configure client-side timeouts based on your workload requirements. A timeout of 60 to 120 seconds is recommended.

- Expect longer processing times when using the heuristic strategy with large datasets.
- Consider asynchronous processing patterns for long-running prediction requests.

Security Considerations

Consider the following security and data handling recommendations:

- The service does not persist query data or prediction results.
- Context data is retrieved for each request from registered Tabular Artifacts.
- Review data residency requirements when selecting deployment regions.

1.5 Optimizations

Optimizations lets you evaluate and refine your generative AI workflows and prompts.

Key features of optimizations include:

- **Evaluations:** Systematically assess and validate AI model performance using evaluation workflows and metrics. Manage the complete lifecycle of your evaluation processes and analyze results to ensure your AI solutions meet quality standards before deployment. For more information, see Evaluations [page 654].
- **Prompt Optimization:** Enhance prompt effectiveness by optimizing templates against target datasets to maximize specified performance metrics. Leverage prompt registry integration and systematic testing to improve model outputs and achieve better alignment with business objectives. For more information, see Prompt Optimization [page 664].
- **Metrics:** Monitor model performance using built-in and custom indicators to ensure your AI applications meet business requirements. For more information, see Metrics [page 684].

1.5.1 Prerequisites

1.5.1.1 Register an Object Store for Optimizations

Procedure

To create an object store secret for your object store, send a POST request to the endpoint `$AI_API_URL/v2/admin/objectStoreSecrets` and include your credentials in the body of your request.

> ### Note
>
> For all storage types **except** Azure Blob Storage, all `<data>` fields are required. For Azure, required fields are specified.

- For Amazon S3:

```
curl --location --request POST "$AI_API_URL/v2/admin/objectStoreSecrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <Resource group>' \
--data-raw '{
        "name": "default",
        "type": "S3",
        "bucket": "<S3 bucket name>",
        "endpoint": "<S3 end point>",
        "pathPrefix": "<A path prefix that follows the bucket name>",
        "region": "<S3 region>",
        "data": {
              "AWS_ACCESS_KEY_ID": "<AWS access key ID>",
              "AWS_SECRET_ACCESS_KEY": "<AWS secret access key>"
        }
    }'
```

- For OSS (Alicloud Object Storage Service):

```
curl --location --request POST "$AI_API_URL/v2/admin/objectStoreSecrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <Resource group>' \
--data-raw '{
        "name": "default",
        "type": "oss",
        "pathPrefix": "<path prefix to be appended with bucketname>",
        "data": {
            "BUCKET": "<bucket-name>",
            "ENDPOINT": "oss-cn-shanghai.aliyuncs.com",
            "REGION": "",
            "ACCESS_KEY_ID": "xxxxx",
            "SECRET_ACCESS_KEY": "xxxxx"
        }
}
```

- For Azure Blob Storage:

```
curl --location --request POST "$AI_API_URL/v2/admin/objectStoreSecrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <Resource group>' \
--data-raw '{
        "name": "default",
        "type": "azure",
        "pathPrefix": "<path prefix to be appended>",
        "data": {
            "CONTAINER_URI": "https://account_name.blob.core.windows.net/
container_name",  //required
            "REGION": "<region name>",                  //optional
            "CLIENT_ID": "<azure client id>",           //optional
            "CLIENT_SECRET": "<azure client secret>",   //optional
            "STORAGE_ACCESS_KEY": "sas_token",          //required
            "TENANT_ID": "azure tenant id",             //optional
            "SUBSCRIPTION_ID": "subscription id",       //optional
    }
}
```

→ Tip

The `pathPrefix` is useful if you share the same bucket for different projects. You can set the name of your project folder to **`my-ml-project1`**, for example. All data is then stored in that folder.

ⓘ Note

If the `AI-Resource-Group` header isn't specified, the `<Resource Group>` is assigned the value `"default"` automatically.

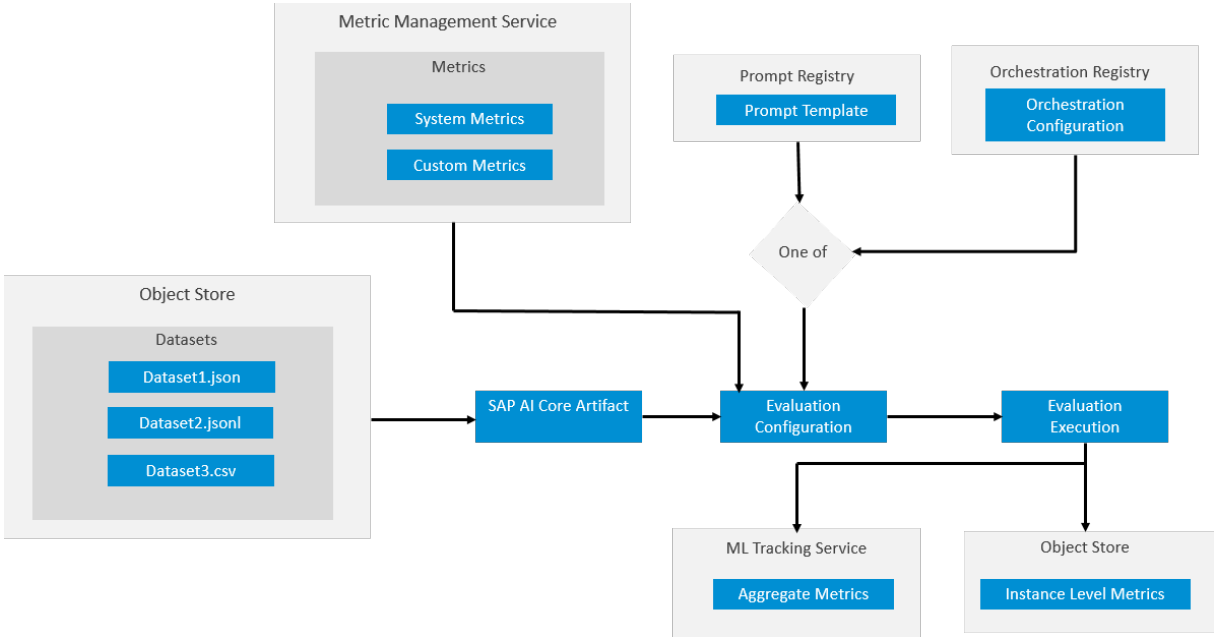
1.5.2 Evaluations

Evaluations provides tools for benchmarking large language models and prompts via orchestration configurations.

Evaluations can be used for the following use cases:

- Evaluating prompt templates and models as orchestration configurations to determine the most effective combination for a use case.
- Use industry-standard predefined metrics for model and prompt comparison with your use-case-specific dataset.
- Use your own custom-defined metrics for your prompt and model evaluations.

Evaluations Overview



1.5.2.1 Prerequisites

1.5.2.1.1 Variable Mapping

Variable mapping enables you to align variable names between orchestration configurations, metric evaluators, and test datasets, ensuring correct data flow even when attribute names differ. Use this feature to resolve naming mismatches and maintain consistency in automated testing and evaluation workflows.

Variable mappings map properties in the orchestration configuration or response and metric evaluators to attributes in the test data set. By default, variable names in the prompt templates are matched against the attribute names in the test dataset. Where variable names and test data attribute names are mismatched, mapping is required. For example, the prompt template in your orchestration configuration contains an attribute called “query” containing your prompt message, whereas the corresponding attribute in your ground truth dataset is called “questions”. In this case, you have to map the orchestration configuration variable “query” to the dataset variable “questions”.

Variable mappings are provided as a JSON dictionary in the following format:

Sample Code

```
{
  "target_prefix/field_name": "source_prefix/field_name"
}
```

Case 1

Mapping a variable in the orchestration workflows to an attribute in the test dataset

In the following example, “prompt” and “data” are target and source prefixes for the orchestration workflow prompt template and the test dataset respectively and “query” and “question” are the name mismatched prompt template variable and dataset attribute.

Sample Code

```
{
  "prompt/query": "data/question"
}
```

Case 2

Mapping a variable of a metric evaluator to an attribute in the test dataset.

For variables that are commonly used in judge prompts, the following system-defined variable names are defined, and their values are populated automatically.

Variable Name	Description
aicore_llm_prompt	The prompt sent to the LLM
aicore_llm_completion	The completion string returned by the LLM

You must define the following common values for variables.

Variable Name	Description
reference	The ground truth reference string provided by the user.

The variable names starting with "aicore" are reserved and must not be used for other variables or dataset attributes.

In the following example, the target prefix is the ID of the metric evaluator and the source prefix is “data”.

Sample Code

```
{
  ...
  "<metric_id>/reference": "data/ground_truth"
  ...
}
```

To map a dataset attribute to a variable being used in all metric evaluators, the target prefix `all_metrics` can be used as the target prefix. For example:

Sample Code

```
{
  ...
  "all_metrics/reference": "data/ground_truth"
  ...
}
```

1.5.2.1.2 Register Evaluation Artifacts

You can evaluate different orchestration configurations and prompt variable values against the shortlisted models to find the best orchestration configuration for your use case.

Prerequisites

- You have an orchestration deployment running. For more information, see Get An Orchestration Deployment URL [page 7] or Create a Deployment for Orchestration [page 703].
- You've registered an object store for Amazon S3, OSS, or Azure Blob Storage. For more information, see Register an Object Store for Optimizations [page 652].

Context

To run an evaluation, you must provide the following artifacts:

- A test dataset containing data needed for the evaluation, such as values for the prompt variables and ground truth references. The following file formats are supported:
  - `CSV`
  - `JSON`
  - `JSONL`

The following folder structure is recommended:

```
root-directory
└── testdata
    ├── dataset1.jsonl
    └── dataset2.csv
```

Procedure

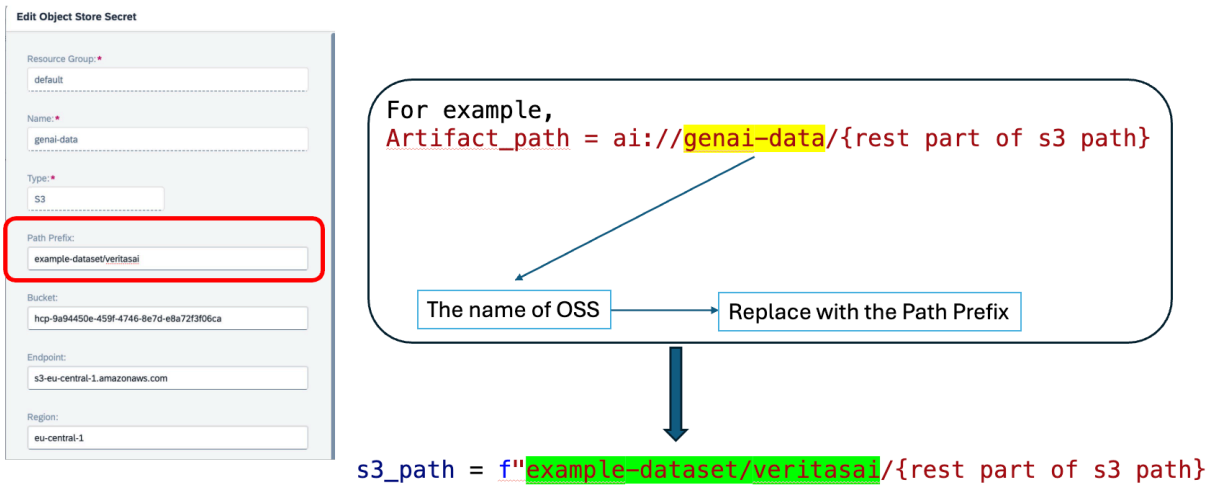
Send a request to the endpoint `{{apiurl}}/v2/lm/artifacts`, and include the prompt evaluation label and the path to your artifact within your object store in your request.

Sample Code

```
curl --location --request POST "{{apiurl}}/v2/lm/artifacts" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <resource group>' \
--data-raw '{
        "name": "yourTestData",
        "kind": "other",
        "url": "ai://<name of object store secret>/<relative path to
evaluations testData folder>"
        "description": "demo artifacts for evaluation flow.",
        "scenarioId": "genai-evaluations"
    }'
```

- The `url` must begin with `ai://`, followed by the name of your object store secret and the relative path to the directory that contains your evaluation files. The directory containing all files for evaluation is registered as a single artifact.
- The `scenarioId` must be `"genai-evaluations"`.

The data path specified in the URL field uses the name of the OSS you created as the path prefix. This replaces the originally assigned path prefix, as shown in the image below.



1.5.2.2 Evaluation Workflow

1.5.2.2.1 Create an Evaluation

Prerequisites

- You have prepared your dataset and registered it as an artifact. For more information, see Register Evaluation Artifacts [page 656].
- You have chosen your metric types. If you're using custom metrics, you've prepared your custom metric. For more information, see System-Defined Evaluation Metrics [page 684] and Custom Metrics [page 695].
- If your prompt template variable names and test data attribute names are mismatched, you have prepared your variable mapping. For more information, see Variable Mapping [page 655].
- You have created a prompt for your evaluation, and your prompt is available in the prompt registry for your evaluation. For more information, see Prompt Registry [page 732].

Create a Configuration

To create a configuration, send a POST request to endpoint `$AI_API_URL/v2/lm/configurations`.

Include the `artifactId` for your registered dataset as `inputArtifactBindings`.

Include the `executableId` `"genai-evaluations-simplified"`

Include the following in your request as `parameterBindings`:

Input Parameter		Description
`datasetFolder`		An AI Core artifact whose path points to the root folder in the object store containing the test dataset for evaluation.
`testDataset`		A JSON containing the path to a test dataset relative to the rootFolder and a file type.
`metrics`		The IDs of metrics to be evaluated. A string containing comma-separated names of system-defined metrics, or IDs of custom metrics.
`promptTemplateScope` (Optional)		Defines the scope of the prompt template. Accepted values are `tenant` (default) or `resourcegroup`. You can omit this field for tenant-level templates. You must set it to resource group when the template is created with the `AI-Resource-Group-Scope: true` header.
`variableMapping` (Optional)		A JSON containing column name mappings. For more information, see Variable Mapping [page 655].
`tags` (Optional)		A JSON containing name-value pairs containing user-defined metadata.
`orchestrationDeploymentURL`		The orchestration deployment to use for orchestration completion and LLM-as-a-judge completion calls.
`testRowCount` (Optional)		Specifies the number of rows to be selected from the testDataset for evaluation.
A reference to your prompt template and models or to your orchestration config	`promptTemplate` and `models` combined.	• The ID of the prompt template stored in the prompt registry. • The models that correspond to the prompt template. A string containing a comma-separated list of models and their versions. For example, `"gpt-4.1:latest, gpt-4o-mini:latest"`.|
	`orchestrationRegistryIds`	The ID of the orchestration config stored in the orchestration registry.

Example with `promptTemplate` and `models` combined

Sample Code

```
curl --request POST "$AI_API_URL/v2/lm/configurations" \
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP" \
--header "Content-Type: application/json" \
--data '{
    "name": "genai-eval-conf",
    "scenarioId": "genai-evaluations",
    "executableId": "genai-evaluations-simplified",
    "inputArtifactBindings": [
        {
            "key": "datasetFolder",
            "artifactId": "<artifactId>"
        }
    ],
    "parameterBindings": [
        {
            "key": "orchestrationDeploymentURL",
            "value": "https://<ai-api-base-url.com>/v2/inference/deployments/
<deploymentId>"
        },
        {
            "key": "tags",
            "value": "{}"
        },
        {
            "key": "variableMapping",
            "value": "{"json_schema_match/json_schema":"data/
json_schema_value"}"
        },
        {
            "key": "metrics",
            "value": "BERT Score,BLEU,ROUGE,"
        },
        {
            "key": "testDataset",
            "value": "{"path":"data/template_variables.csv", "type": "csv"}"
        },
        {
            "key": "promptTemplate",
            "value": "<promptTemplateId>"
        },
        {
            "key": "promptTemplateScope",
            "value": "resourcegroup"
        },
        {
            "key": "models",
            "value": "<model>:latest, <model>:latest"
        },
    ]
}'
```

Example with `orchestrationRegistryIds`

Sample Code

```
curl --request POST "$AI_API_URL/v2/lm/configurations" \
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP" \
--header "Content-Type: application/json" \
--data '{
```

```
    "name": "genai-eval-conf",
    "scenarioId": "genai-evaluations",
    "executableId": "genai-evaluations-simplified",
    "inputArtifactBindings": [
        {
            "key": "datasetFolder",
            "artifactId": "<artifactId>"
        }
    ],
    "parameterBindings": [
        {
            "key": "orchestrationDeploymentURL",
            "value": "https://<ai-api-base-url.com>/v2/inference/deployments/
<deploymentId>"
        },
        {
            "key": "tags",
            "value": "{}"
        },
        {
            "key": "variableMapping",
            "value": "{"json_schema_match/json_schema":"data/
json_schema_value"}"
        },
        {
            "key": "metrics",
            "value": "BERT Score,BLEU,ROUGE,"
        },
        {
            "key": "testDataset",
            "value": "{"path":"data/template_variables.csv", "type": "csv"}"
        },
          { "key": "orchestrationRegistryIds",
            "value": "<orchestrationConfigId>" },
    ]
}'
```

You will receive a configuration ID. Note the ID for the next step.

Start an Evaluation

You can start an evaluation by creating an execution for your configuration.

Evaluation Job Progress Estimation

When you view the status of your execution, the results include progress details that are normalized to N/100 format. For example: “3/5” is normalized to “3/100”. Completed executions always show “100/100”

> **Note**
>
> Pod-level progress is not granular: If an execution spawns 3 pods, progress will jump through: “0/100” → “1/100” → “2/100” → “3/100” → “100/100”.

Additionally, you can implement custom progress reporting in your workflow code by including an `ARGO_PROGRESS_FILE`. For more information, see Argo Progress File .

To start an evaluation, send a POST request to endpoint `$AI_API_URL/v2/lm/executions`: Include your configuration ID in your request.

For example:

Sample Code

```
curl --request POST "$AI_API_URL/v2/lm/executions"
--header "Authorization: Bearer $TOKEN"
--header "AI-Resource-Group: $RESOURCE_GROUP"
--header "Content-Type: application/json" \
--data '{
"configurationId": "<configurationID>"
}'
```

Results

Your evaluation will take some time to run. To check the status of an evaluation, send a GET request to endpoint `{{apiurl}}/v2/lm/executions/{{executionid}}`.

Next Steps

The evaluation outputs the following:

- An SQLite DB file which stores the orchestration input, orchestration output, values for all the metrics calculated for this orchestration output and statistics such as latency for this orchestration output. These metric values are called raw metric values. This SQLite DB file is stored in the object store as an AI Core output artifact.
- A set of metrics whose values are aggregated from the raw metric values. The aggregate metrics are stored in the tracking service. The user-defined tags along with the run names are stored with the metrics. Post evaluation completion the runs generated by the workload along with the aggregate metrics can be seen by calling the tracking api.

When the evaluation is complete, you can view and analyze your results. For more information, see View Evaluation Results [page 662].

After and evaluation is complete, there may be residual output artifacts such as logs or instance level metrics, that can be safely deleted by users.

1.5.2.3 View Evaluation Results

Prerequisites

You have successfully run an evaluation execution. For more information, see Create an Evaluation (Simplified) [page 658].

Context

The evaluation job will calculate and report the aggregate job statistics for each completion.

Aggregate job statistics include the following:

Statistic	Description
average_latency	The average time taken in seconds to get a completion from the orchestration service
completion_count	Number of completions evaluated
total_prompt_tokens	Sum of prompt_tokens of completions
total_completion_tokens	Sum of completion_tokens of completions

The following aggregate result statistics are included for all metrics:

Statistic	Description
mean	The center point of a collection of values, also known as arithmitic mean
stddev	The amount of deviation from the mean
median	The middle value or 50th percentile of a collection of values
p90	The value at the 90th percentile of a collection of values
p95	The value at the 95th percentile of a collection of values

In addition, `BERT Score` includes these statistics for the sub metrics F1, precision, and recall.

Recall is the fraction of correctly labelled positive instances, in the context of all available positive instances. It can be computed using the equation: Recall = TP/(TP + FN), where TP is the number of true positives and FN is the number of false negatives.

Precision is the fraction of correctly labeled positive instances, in the context of all positively labelled instances. It is computed using the equation: Precision = TP/(TP + FP), where TP is the number of true positives and FP is the number of false positives.

The **F1 score** is the harmonic mean of the precision and recall. It can be computed with the equation: F1 = 2 (precision*recall)/(precision + recall)

Procedure

Retrieve your results by sending a GET request to one of the following endpoints:

- Using your `executionId`: `{{apiurl}}/v2/lm/metrics?tagFilters=evaluation.ai.sap.com/child-of={execution_id}`
- Using your `runName`: `{{apiurl}}/v2/lm/metrics?tagFilters=evaluation.ai.sap.com/run-name={run_name}`

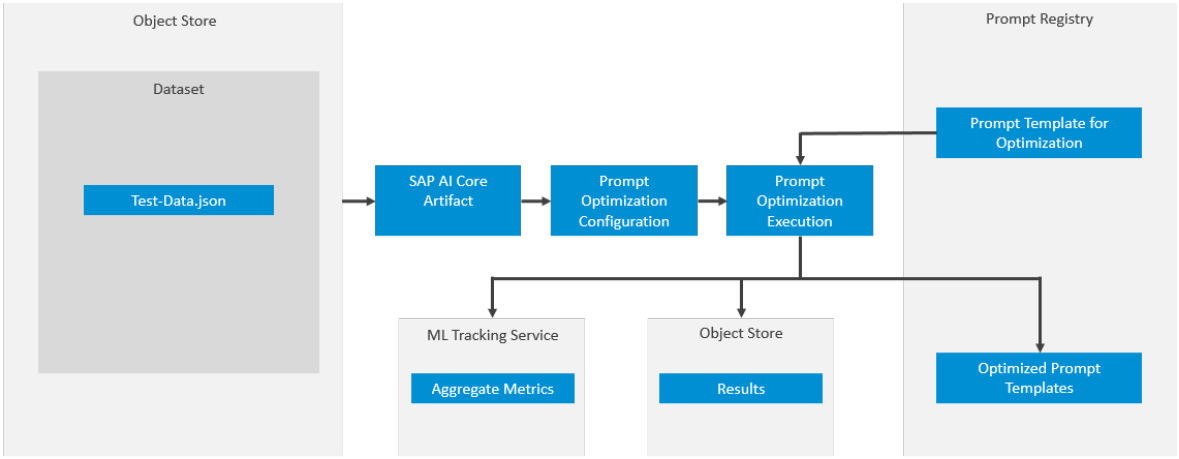
> **Note**
>
> After and evaluation is complete, there may be residual output artifacts such as logs or instance level metrics, that can be safely deleted by users.

1.5.3 Prompt Optimization

Prompt Optimization utilizes an input prompt template from the prompt registry and a JSON dataset in the object store containing desirable responses. It optimizes the input prompt to maximize a specified metric, with metrics tracked in the ML Tracking Service. The resulting optimized prompt is saved back to the prompt registry, and additional result information is stored in the object store. This optimization process is model-specific to accommodate different model behaviors and is executed as a distinct run.

> **Note**
>
> Prompt optimization executions can take from minutes to multiple hours to run, and will submit a large number of prompt requests to the target models.

> **Restriction**
>
> Mistral and DeepSeek models are not supported for prompt optimization.



Prompt Optimization Workflow

The workflow for prompt optimization is as follows:

1. Register your object store secrets. For more information, see Register an Object Store for Optimizations [page 652].
2. Prepare a prompt template. For more information, see Template Preparation [page 665].
3. Prepare an optimization dataset. For more information, see Dataset Preparation [page 666].
4. Register your optimization dataset as a prompt optimization artifact. For more information, see Register Optimization Artifacts [page 667].
5. Create a configuration for your prompt optimization. For more information, see Create a Configuration for a Prompt Optimization [page 669]

   > **Note**
   >
   > Optimization calls often require multiple repetitions to an LLM, with each repetition incurring costs. Additionally, some calls may involve extra internal processes, leading to actual costs that differ from those predicted by cost calculators.
   >
   > For more information, see the Cost Calculator.

6. Create an execution for your configuration. For more information, see Create an Execution for your Prompt Optimization [page 676].

1.5.3.1 Prerequisites

1.5.3.1.1 Template Preparation

You prepare a prompt template and make it available for optimization by storing it in the prompt registry. For more information, see Prompt Registry [page 732].

Your prompt template must meet the following conditions:

- It is available in the prompt registry
- The use prompt must contain one or more placeholders
- The system prompt must not contain variables.

For example:

> **Sample Code**

```
---
{{?input}}
---
Extract and return a json with the following keys and values: - "urgency" as
one of `high`, `medium`, `low` -
"sentiment" as one of `negative`, `neutral`, `positive` - "categories" Create
a dictionary with categories as
keys and boolean values (True/False), where the value indicates whether the
category is one of the best
matching support category tags from: `emergency_repair_services`,
`routine_maintenance_requests`,
```

```
`quality_and_safety_concerns`, `specialized_cleaning_services`,
`general_inquiries`,
`sustainability_and_environmental_practices`,
`training_and_support_requests`, `cleaning_services_scheduling`,
`customer_feedback_and_complaints`, `facility_management_issues` Your
complete message should be a valid json
string that can be read directly and only contain the keys mentioned in the
list above. Never enclose it in
```json...```, no newlines, no unnecessary whitespaces.
```

1.5.3.1.2 Dataset Preparation

The dataset shows the desirable outcome of an input prompt and corresponding values for placeholders in the input prompt.

Your dataset must meet the following conditions:

- It has a minimum of 25 and maximum of 200 samples
- It contains the exact placeholder names that are used in the template in the `fields` object.

> ⚠ Caution
>
> Do not include confidential of personally identifiable information in the dataset.

For example:

Sample Code

```
[
    {
        "fields": {
            "input": "Subject: Urgent Assistance Required for Specialized
Cleaning Services\n\nDear ProCare
            Facility Solutions Support Team. Could you please arrange for a
specialized cleaning team to
            visit our home at the earliest convenience? We would greatly
appreciate it if this could be
            prioritized since we want to host a large party this
week.\n\nThank you for your prompt
            attention to this matter. We look forward to your swift response
and assistance.\n\nBest
            regards,\n[Sender]"
        },
        "answer": "{\"categories\": {\"routine_maintenance_requests\": false,
        \"customer_feedback_and_complaints\": false,
\"training_and_support_requests\": false,
        \"quality_and_safety_concerns\": false,
\"sustainability_and_environmental_practices\": false,
        \"cleaning_services_scheduling\": false,
\"specialized_cleaning_services\": true,
        \"emergency_repair_services\": false, \"facility_management_issues\":
false,
        \"general_inquiries\": false}, \"sentiment\": \"neutral\",
\"urgency\": \"high\"}"
    },
  {...}
]
```
```

1.5.3.1.3 Register Optimization Artifacts

Prerequisites

- You've registered an object store for Amazon S3, OSS, or Azure Blob Storage. For more information, see Register an Object Store for Optimizations [page 652].
- You've prepared a dataset in JSON format and stored it in your registered object store. For more information, see Dataset Preparation [page 666].

Context

To run an optimization, you must provide a test dataset in JSON format. Your test dataset must contain data needed for the optimization, such as values for the prompt variables and ground truth references.

Use the following folder structure:

```
root-directory
└── testdata
    ├── dataset1.json
```

If you want to use train and test datasets, use the following folder structure.

```
root-directory
└── data
    ├── train-dataset.json
    ├── test-dataset.json
```

All optimization-relevant files must be stored under file paths relative to the URL of the registered artifact.

Procedure

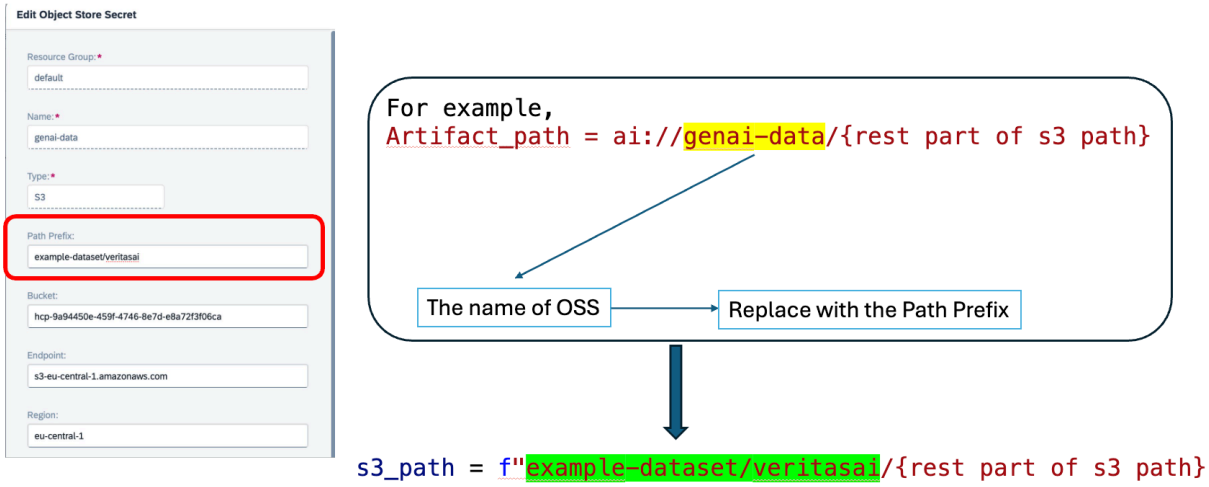
Send a POST request to the endpoint `$AI_API_URL/v2/lm/artifacts`, and include the prompt optimization label and the path to your artifact within your object store in your request.

Sample Code

```
curl --location --request POST "$AI_API_URL/v2/lm/artifacts" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <resource group>' \
--data-raw '{
        "name": "yourTestData",
        "kind": "dataset",
        "url": "ai://<name of object store secret>/<relative path to
optimizations testData folder>"
        "description": "demo artifacts for optimization flow.",
        "scenarioId": "genai-optimizations"
    }'
```

- The `url` must begin with `ai://`, followed by the name of your object store secret and the relative path to the directory that contains your optimization files. The directory containing all files for optimization is registered as a single artifact.
- The `scenarioId` must be `"genai-optimizations"`.

The data path specified in the URL field uses the name of the OSS you created as the path prefix. This replaces the originally assigned path prefix, as shown in the image below.



Next Steps

Create a configuration for your optimization. For more information, see Create a Configuration for a Prompt Optimization [page 669].

1.5.3.1.4 Variable Mapping

Variable mapping enables you to align variable names between prompts and test datasets, ensuring correct data flow even when attribute names differ. Use this feature to resolve naming mismatches and maintain consistency in automated prompt optimization workflows.

Variable mappings map properties in the prompt optimization configuration to attributes in the test data set. By default, variable names in the prompt templates are matched against the attribute names in the test dataset. Where variable names and test data attribute names are mismatched, mapping is required.

In the following example, the keys `user_query` and `context` reference items in a prompt, and values `input_text` and `background_info` reference dataset fields. These are mapped through key-value pairs.

Variable mappings are provided as a JSON dictionary in the following format:

```
{"user_query": "input_text", "context": "background_info"}
```

1.5.3.2 Create a Configuration for a Prompt Optimization

Configuration for a prompt optimization defines the parameters, artifacts, and model settings that SAP AI Core uses to execute a prompt optimization.

Prerequisites

- You're using the `extended` service plan. For more information, see Service Plans.
- You've registered an object store secret for input artifacts. For more information, see Register an Object Store for Optimizations [page 652].
- You've prepared an optimization dataset and registered it as an optimization artifact. For more information, see Dataset Preparation [page 666] and Register Optimization Artifacts [page 667].
- You've prepared a prompt template, and your template is available in the prompt registry. For more information, see Template Preparation [page 665].
- If you're using custom metrics, you've prepared your custom metric. For more information, see Create a Custom Metric [page 697].
- If your prompt template variable names and test data attribute names are mismatched, you have prepared your variable mapping. For more information, see Variable Mapping [page 668].

> ⚠ Restriction
>
> Only custom metrics with the evaluation method LLM-as-a-judge and numerical or Boolean output types can be included in prompt optimizations.

Procedure

Create a configuration by sending a POST request to endpoint `$AI_API_URL/v2/lm/configurations`.

Include the following in your request:

Field	Value
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
$RESOURCE_GROUP	The AI resource group assigned to your account
$TOKEN	Your access token for SAP AI Core
`inputArtifactBindings`	The `artifactId` for your registered dataset
`scenarioId`	`genai-optimizations`
`executableId`	`genai-optimizations`

Field	Value		
`parameterBindings`	`basePrompt` (mandatory)		Reference to the prompt in the prompt registry that should be optimized in the format "`<scenario>/<promptName>:<promptVersion>`"
	`baseModel` (optional)		In case an evaluation of the selected metric should be run against a baseline model without optimization, to determine the pre-optimization metric. Default: `none`.
	`modelParams` (optional)		A JSON string specifying the model that the parameters should be applied to, and a comma separated list of parameters and parameter values. The following parameters are supported: • `temparature` (0-1) • `max_tokens` (0-n (int)) Any unsupported variables passed will be ignored.
	A dataset selection. Choose from the following: • `dataset`: a single dataset that will be split into test and train • A combination of `trainDataset` and `testDataset`: a dataset that you have split yourself into test and train, referenced in two parts. **Note** The maximum number of samples across all datasets in an optimization is 200.	`dataset`	The dataset that is used as basis for optimization. The dataset contains desired responses from the target model. The optimization will maximize the optimization metric by varying the input prompt. This parameter specifies the path to the JSON contained in the input artifact that is submitted as part of the configuration. The dataset must contain a minimum of 25 samples.
		`trainDataset` (obligatory if `testDataset` is selected)	The train dataset used for optimization. The file must be in JSON format and stored in the input artifact. Use this field when you want to provide a dedicated dataset for training instead of using a single dataset. The dataset must contain a minimum of 25 samples.

Field	Value		
		`testDataset` (obligatory if `trainDataset` is selected)	The test dataset used to evaluate optimization performance. The file must be in JSON format and stored in the input artifact. Use this field when you provide train and test datasets. The dataset must contain a minimum of 1 sample.
	`promptTemplateScope` (Optional)		Defines the scope of the prompt template. Accepted values are `tenant` (default) or `resourcegroup`. You can omit this field for tenant-level templates. You must set it to resource group when the template is created with the `AI-Resource-Group-Scope: true` header.

Field	Value	
	A metric reference (mandatory) The metric to be used for optimizing. You can choose from: • System defined optimization metrics To use a system defined optimization metric, include `optimizationMetric`. • Your own custom metric. For more information, see Custom Metrics [page 695]. To use a custom metric, include `customMetricId`.	The possible system defined optimization metrics are: • `LLMaaJ:Sem_Sim_10` This metric uses an LLM-as-a-judge to evaluate the semantic similarity of the model response relative to the target golden dataset, on a scale of 1-10. • `LLMaaJ:Sem_Sim_1` This metric uses an LLM-as-a-judge to evaluate the semantic similarity of the model response relative to the target golden dataset, and returns a binary response. • `JSON_Match` This metric determines if the LLM's JSON output matches the dataset output. The precision, recall, and f1 scores are computed based on individual fields in the JSON and averaged across all samples in the dataset. This is useful for applications that require structured outputs. • `FC:ExactMatch` A variation on `ExactMatch` specifically for use cases that include function calling. • `BLEU` This metric evaluates machine-translated text quality by calculating n-gram precision between candidate and reference translations.

Field	Value	
		Scores range from 0 to 1, with higher values indicating greater similarity. • ROUGE A set of metrics for evaluating summarization and machine translation by measuring overlap in n-grams, word sequences, and word pairs between generated and reference texts. This implementation is case-insensitive. • METEOR This metric evaluates response quality using a combination of precision, recall, and synonym matching. Better handles morphological variations and paraphrasing compared to BLEU. Particularly useful for languages with rich morphology. • EXACT_MATCH A Boolean indicating whether the output exactly matches the reference.
		The ID of your chosen custom metric. **⚠ Restriction** Only custom metrics with the evaluation method LLM-as-a-judge **and** numerical **or** Boolean output types can be included in prompt optimizations.

Field	Value	
	`targetModels` (mandatory)	A comma separated list of models with version for which to optimize the prompt. A prompt can be optimized for a maximum of four models. The format is `<modelName>:<modelVersion>`. The target models must be available in the corresponding region. For more information, see SAP Note 3437766.
	`targetPromptMapping` (mandatory)	Provides a mapping to the optimized prompt for storage in the prompt registry in the format `<modelName>:<modelVersion>=<promptName>:<promptVersion>`.
	prototypeMode (optional)	`true` or `false` (default). Prototype mode is not recommended for productive use. For reliable and generalizable results, a larger sample size (25+ samples) is needed, and standard mode is recommended. This mode can be used when you don't have enough data to build a proof-of-concept but want to use a minimum of 3 samples for prototyping. Number of samples: • **Dataset**: Your dataset must contain a minimum of 3 samples. • **Split Dataset**: Your train dataset must contain a minimum of 3 samples, and your test dataset must contain a minimum of 1 sample. Datasets containing more than 200 samples will cause the jobs to fail.

Field	Value	
	`variableMapping` (Optional)	A JSON containing column name mappings. For more information, see Variable Mapping [page 668].
`includeFewShotExamples` (optional)	Whether to include some parts of the provided dataset in the input prompt to improve the output. Default: `false`.	

For example:

Sample Code

```
curl --request POST "$AI_API_URL/v2/lm/configurations" \
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP" \
--header "Content-Type: application/json" \
--data '{
    "name": "genai-eval-conf",
    "scenarioId": "genai-optimizations ",
    "executableId": "genai-optimizations ",
    "inputArtifactBindings": [
        {
            "key": "datasetFolder",
            "artifactId": "1f50fbf8-18d6-4e72-ae10-04fdbc087815"
        }
    ],
    "parameterBindings": [
        {
            "key": "basePrompt",
            "value": "<scenario>/<promptName>:<promptVersion>"
        },
        {
            "key": "modelParams",
            "value": "{"<modelName1>:<modelVersion1>":
{"temperature":0.7,"max_tokens":1000},"<modelName2>:<modelVersion2>":
{"temperature":0.9}}"
        },
        {
            "key": "dataset",
            "value": "data.json"
        {
            "key": "optimizationMetric",
            "value": "JSON_Match”
        },
        {
            "key": "variableMapping",
            "value": "{"<prompt_variable1>":"<data_variable1>"}"
        },
        {
            "key": "promptTemplateScope",
            "value": "resourcegroup"
        },
        {
            "key": "targetModels",
            "value": "<modelName>:<modelVersion>”
        },
        {
            "key": "targetPromptMapping",
            "value": “<modelName>:<modelversion>=<promptName>:<promptVersion>”
        },
    ]
```

```
}'
```

> **Note**
>
> To use `trainDataset` and `testDataset`, replace the key and value attributes in the `dataset` as shown below.

```
{
            "key": "trainDataset",
            "value": "train-dataset.json",
          }
          {
            "key": "testDataset",
            "value": "test-dataset.json",
          }
```

1.5.3.3 Create an Execution for your Prompt Optimization

Prerequisites

You've created a configuration for your optimization. For more information, see Create a Configuration for a Prompt Optimization [page 669].

Procedure

To create an execution, send a POST request to endpoint `$AI_API_URL/v2/lm/executions`. Include your configuration ID in your request.

Field	Value
$AI_API_URL	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
$RESOURCE_GROUP	The AI resource group assigned to your account
$TOKEN	Your access token for SAP AI Core
`configurationId`	Your `configurationId` from the previous step

For example:

> **Sample Code**

```
curl --request POST "$AI_API_URL/v2/lm/executions"
--header "Authorization: Bearer $TOKEN"
```

```
--header "AI-Resource-Group: $RESOURCE_GROUP"
--header "Content-Type: application/json" \
--data '{"configurationId": "<YourConfigurationId>"}'
```

Results

Your execution will take some time to run.

You can check the status of the execution, by sending a GET request to endpoint `$AI_API_URL/v2/lm/executions/<executableId>`.

After the execution has reaches `COMPLETED` status, you can check your results.

The prompt optimization execution outputs the following:

- A set or metrics that are stored in the tracking service. Post execution completion the metrics can be seen by calling the tracking api. For more information, see Get Metrics.
- A set of optimized prompt templates in the prompt registry, one per model. In case the optimization didn't give better results than the original prompt, no prompt template is stored. You can review the optimized template in the prompt registry. For more information, see Get a Prompt Template [page 737].

1.5.3.4 Response Formatting

You can define a schema for the response structure to follow. To do this, include a `response_format` field in your prompt template and populating it with details of your structure. `response_format` is an optional field.

For example:

Sample Code

```
{
    "name": "response-format-prompt-template",
    "version": "0.0.1",
    "scenario": "genai-optimizations",
    "spec": {
        "template": [
            {
                "role": "system",
                "content": "you are a helpful assistant"
            },
            {
                "role": "user",
                "content": "Giving the following message:\n---\n{{?input}}
\n---\nExtract and return a json with the follwoing keys and values:\n-
\"urgency\" as one of `high`, `medium`, `low`\n- \"sentiment\" as one of
`negative`, `neutral`, `positive`\n- \"categories\" Create a dictionary with
categories as keys and boolean values (True/False), \n  where the value
indicates whether the category is one of the best matching support category
tags from: \n  `emergency_repair_services`, `routine_maintenance_requests`,
`quality_and_safety_concerns`, \n  `specialized_cleaning_services`,
`general_inquiries`, `sustainability_and_environmental_practices`,
\n  `training_and_support_requests`, `cleaning_services_scheduling`,
`customer_feedback_and_complaints`, \n  `facility_management_issues`\nYour
```

```
complete message should be a valid json string that can be read directly
and only contain \nthe keys mentioned in the list above. Never enclose it in
```json...```, no newlines, no unnessacary whitespaces."
            }
        ],
        "response_format": {
            "type": "json_schema",
            "json_schema": {
                "name": "facility_output",
                "strict": true,
                "schema": {
                    "type": "object",
                    "properties": {
                        "urgency": {
                            "type": "string",
                            "description": "The urgency of the request",
                            "enum": [
                                "high",
                                "medium",
                                "low"
                            ]
                        },
                        "sentiment": {
                            "type": "string",
                            "description": "The sentiment of the request",
                            "enum": [
                                "positive",
                                "negative",
                                "neutral"
                            ]
                        },
                        "categories": {
                            "type": "object",
                            "properties": {
                                "emergency_repair_services": {
                                    "type": "boolean"
                                },
                                "routine_maintenance_requests": {
                                    "type": "boolean"
                                },
                                "quality_and_safety_concerns": {
                                    "type": "boolean"
                                },
                                "specialized_cleaning_services": {
                                    "type": "boolean"
                                },
                                "general_inquiries": {
                                    "type": "boolean"
                                },
                                "sustainability_and_environmental_practices":
{
                                    "type": "boolean"
                                },
                                "training_and_support_requests": {
                                    "type": "boolean"
                                },
                                "cleaning_services_scheduling": {
                                    "type": "boolean"
                                },
                                "customer_feedback_and_complaints": {
                                    "type": "boolean"
                                },
                                "facility_management_issues": {
                                    "type": "boolean"
                                }
                            },
                            "required": [
                                "emergency_repair_services",
```
```

```
                                "routine_maintenance_requests",
                                "quality_and_safety_concerns",
                                "specialized_cleaning_services",
                                "general_inquiries",
                                "sustainability_and_environmental_practices",
                                "training_and_support_requests",
                                "cleaning_services_scheduling",
                                "customer_feedback_and_complaints",
                                "facility_management_issues"
                            ],
                            "additionalProperties": false
                        }
                    },
                    "required": [
                        "urgency",
                        "sentiment",
                        "categories"
                    ],
                    "additionalProperties": false
                }
            }
        },
        "additional_fields": {
            "modelParams": {
                "temperature": 0.7,
                "max_tokens": 100
            },
            "modelGroup": "chat"
        }
    }
}
```

Output Code

```
response_format = {
            "type": "json_schema",
            "json_schema": {
                "name": "facility_output",
                "strict": True,
                "schema": {
                    "type": "object",
                    "properties": {
                        "urgency": {
                            "type": "string",
                            "description": "The urgency of the request",
                            "enum": ["high", "medium", "low"]
                        },
                        "sentiment": {
                            "type": "string",
                            "description": "The sentiment of the request",
                            "enum": ["positive", "negative", "neutral"]
                        },
                        "categories": {
                            "type": "object",
                            "properties": {
                                "emergency_repair_services": {"type":
"boolean"},
                                "routine_maintenance_requests": {"type":
"boolean"},
                                "quality_and_safety_concerns": {"type":
"boolean"},
                                "specialized_cleaning_services": {"type":
"boolean"},
                                "general_inquiries": {"type": "boolean"},
                                "sustainability_and_environmental_practices":
{"type": "boolean"},
```

```
                                "training_and_support_requests": {"type":
"boolean"},
                                "cleaning_services_scheduling": {"type":
"boolean"},
                                "customer_feedback_and_complaints": {"type":
"boolean"},
                                "facility_management_issues": {"type":
"boolean"}
                            },
                            "required": [
                                "emergency_repair_services",
"routine_maintenance_requests",
                                "quality_and_safety_concerns",
"specialized_cleaning_services",
                                "general_inquiries",
"sustainability_and_environmental_practices",
                                "training_and_support_requests",
"cleaning_services_scheduling",
                                "customer_feedback_and_complaints",
"facility_management_issues"
                            ],
                            "additionalProperties": False
                        }
                    },
                    "required": ["urgency", "sentiment", "categories"],
                    "additionalProperties": False
                }
            }
        }
```

Evaluating Response Formatting

You can specify which evaluation metric should be used for each field in your response by including the optional parameter `field_evaluation_metrics` in your prompt optimization configuration. This is useful when you have multiple fields with differing response types, where a one size fits all approach does not apply. For example:

Sample Code

```
field_evaluation_metrics = {
    "urgency":"ExactMatch",
    "sentiment":"LLMaaJ:Sem_Sim_1",
    "categories.emergency_repair_services":"ExactMatch"
}
```

For more information about available metrics, see Create a Configuration for a Prompt Optimization [page 669].

1.5.3.5 Tool Calling

You can define a tools for the model to call to perform specific tasks. To do this, include a `tools` field in your prompt template and populating it with details of your tool. `tools` is an optional field. Tools will only be called if the model finds them relevant to answer the prompt.

For example:

Sample Code

```
{
    "name": "tools-prompt-template",
    "version": "0.0.1",
    "scenario": "genai-optimizations",
    "spec": {
        "template": [
            {
                "role": "system",
                "content": "you are a helpful assistant"
            },
            {
                "role": "user",
                "content": "{{?question}}"
            }
        ],
        "tools": [
            {
                "type": "function",
                "function": {
                    "name": "calculate_distance",
                    "description": "Calculate the distance between two
locations.",
                    "strict": true,
                    "parameters": {
                        "type": "object",
                        "properties": {
                            "origin": {
                                "type": "string",
                                "description": "The starting location (city
or address)."
                            },
                            "destination": {
                                "type": "string",
                                "description": "The ending location (city or
address)."
                            },
                            "unit": {
                                "type": "string",
                                "description": "Distance unit. Options: 'km',
'miles'. Default 'km'."
                            }
                        },
                        "required": [
                            "origin",
                            "destination",
                            "unit"
                        ],
                        "additionalProperties": false
                    }
                }
            },
            {
                "type": "function",
                "function": {
                    "name": "restaurant_search",
                    "description": "Find restaurants matching specific
criteria in a given location.",
                    "strict": true,
                    "parameters": {
                        "type": "object",
                        "properties": {
                            "location": {
                                "type": "string",
```

```
                                    "description": "The city or area to search
in."
                                },
                                "cuisine": {
                                    "type": "string",
                                    "description": "The type of cuisine, e.g.
'Italian', 'Chinese', 'Mexican'."
                                },
                                "price_range": {
                                    "type": "string",
                                    "description": "Price tier. Options:
'budget', 'moderate', 'fine_dining'. Default 'moderate'."
                                },
                                "min_rating": {
                                    "type": "number",
                                    "description": "Minimum star rating
(0.0-5.0). Default 0.0."
                                }
                            },
                            "required": [
                                "location",
                                "cuisine",
                                "price_range",
                                "min_rating"
                            ],
                            "additionalProperties": false
                        }
                    }
                },
                {
                    "type": "function",
                    "function": {
                        "name": "weather_forecast",
                        "description": "Get the weather forecast for a location
over a specified number of days.",
                        "parameters": {
                            "type": "dict",
                            "properties": {
                                "location": {
                                    "type": "string",
                                    "description": "The city to get the forecast
for."
                                },
                                "days": {
                                    "type": "integer",
                                    "description": "Number of days to forecast
(1-14)."
                                },
                                "units": {
                                    "type": "string",
                                    "description": "Temperature units. Options:
'celsius', 'fahrenheit'. Default 'celsius'."
                                }
                            },
                            "required": [
                                "location",
                                "days",
                                "units"
                            ],
                            "additionalProperties": false
                        }
                    }
                },
                {
                    "type": "function",
                    "function": {
                        "name": "currency_conversion",
```

```
                    "description": "Convert an amount from one currency to
another using current exchange rates.",
                    "parameters": {
                        "type": "object",
                        "properties": {
                            "amount": {
                                "type": "number",
                                "description": "The amount of money to
convert."
                            },
                            "from_currency": {
                                "type": "string",
                                "description": "The source currency code,
e.g. 'USD', 'EUR', 'GBP'."
                            },
                            "to_currency": {
                                "type": "string",
                                "description": "The target currency code,
e.g. 'USD', 'EUR', 'GBP'."
                            }
                        },
                        "required": [
                            "amount",
                            "from_currency",
                            "to_currency"
                        ],
                        "additionalProperties": false
                    }
                }
            },
            {
                "type": "function",
                "function": {
                    "name": "grocery_store",
                    "description": "Find nearby grocery stores or check
product availability and pricing.",
                    "parameters": {
                        "type": "object",
                        "properties": {
                            "location": {
                                "type": "string",
                                "description": "The city or area to search
in."
                            },
                            "product": {
                                "type": "string",
                                "description": "A specific product to check
availability for."
                            },
                            "organic_only": {
                                "type": "boolean",
                                "description": "Whether to filter for organic
products only. Default false."
                            }
                        },
                        "required": [
                            "location",
                            "product",
                            "organic_only"
                        ],
                        "additionalProperties": false
                    }
                }
            }
        ],
        "additional_fields": {
            "modelParams": {
                "temperature": 0.7,
```

```
                "max_tokens": 100
            },
            "modelGroup": "chat"
        }
    }
}
```

1.5.4 Metrics

Metrics are quantitative measures used to evaluate the performance of machine generative AI models within SAP AI Core. By tracking and analyzing metrics, you can assess model quality, compare different model versions, and ensure that your prompts generate responses that meet your standards and criteria.

Key features include:

- **System-Defined Metrics:** SAP AI Core offers a set of predefined evaluation methods and metrics tailored for common LLM tasks, such as text generation, classification, and summarization. System-defined metrics ensure consistency and comparability across different models and experiments.
  For more information, see System-Defined Evaluation Metrics [page 684].
- **Custom Metrics:** In addition to system-defined metrics, you can define and register your own custom metrics to address specific evaluation needs. This flexibility allows you to tailor performance measurement to your unique use cases and business requirements.
  For more information, see Custom Metrics [page 695].
- **Metric Management:** SAP AI Core provides tools for managing the entire lifecycle of metrics, including registration, versioning, and deprecation. You can organize metrics, track their usage, and ensure that the most relevant metrics are applied to your LLM evaluations.
  For more information, see Metric Management [page 696].
- **Integration with Evaluations:** Metrics are seamlessly integrated into the evaluation workflows. When you run an evaluation job, you can specify which metrics to compute, and the results are automatically stored and made available for analysis.
  For more information, see Evaluations [page 654].
- **Analysis:** Metric results can be compared across different model runs, making it easy to identify trends, detect regressions, and select the best-performing models.

Next Steps

Complete the prerequisites. For more information, see Register an Object Store for Optimizations [page 652].

You can create a custom metric. For more information, see Custom Metrics [page 695].

1.5.4.1 System-Defined Evaluation Metrics

System defined metrics are available when using the `genai-evaluation` global scenario.

The following types of metrics are supported:

- Computed
- LLM-as-a-judge

Computed Metrics

The following system-defined computed metrics are supported, depending on whether you use the simplified or standard evaluation workflow:

Metric	Description	Reference Required
`BERT Score`	BERTScore is a metric for evaluating the quality of text generation by comparing candidate text to reference text. It uses a pre-trained BERT model to generate contextual embeddings for each token, then measures similarity between these embeddings to assess how closely the candidate aligns with the references.	Yes
`BLEU`	BLEU (Bilingual Evaluation Understudy) evaluates machine-translated text quality by calculating n-gram precision between candidate and reference translations. Scores range from 0 to 1, with higher values indicating greater similarity.	Yes
`ROUGE`	ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics for evaluating summarization and machine translation by measuring overlap in n-grams, word sequences, and word pairs between generated and reference texts. This implementation is case-insensitive.	Yes
`JSON Schema Match`	Validates an LLM-generated response against a predefined JSON Schema, returning a boolean indicating whether the response matches the schema. For more information, see JSON Schema Match [page 691].	No

Metric	Description	Reference Required
`Content Filter on Input`	Boolean indicating whether the input content filter was invoked.	No
`Content Filter on Output`	Boolean indicating whether the output content filter was invoked.	No
`Exact Match`	Boolean indicating whether the output exactly matches the reference.	Yes
`Language Match`	Indicates whether the output matches the expected language. Results may be inaccurate when the text is very short. For more information, see Language Match [page 688].	No

LLM-as-a-Judge Metrics

The following system-defined LLM-as-a-judge metrics are supported, depending on whether you use the simplified or standard evaluation workflow:

Metric	Description	Reference Required
`Pointwise Instruction Following`	Evaluates the model's ability to follow the instructions provided in the user prompt. Scores range from 1 to 5, with 1 indicating no fulfillment and 5 indicating complete fulfillment.	No
`Pointwise Correctness`	Evaluates whether an LLM response is correct, accurate, and factual using a user-provided reference, for both general and retrieval-augmented (RAG) use cases. Scores range from 1 to 5, with 1 indicating completely incorrect and 5 indicating fully correct	No
`Pointwise Answer Relevance`	Measures how closely the model's response relates to the user prompt, for both general and RAG use cases. Scores range from 1 to 5, with higher values indicating greater relevance.	No

Metric	Description	Reference Required
`Pointwise Conciseness`	Measures how short and concise the model's response is. Scores range from 1 to 5, with higher values indicating a more concise answer.	No
`Pointwise RAG Groundedness`	Evaluates whether the model's response is supported by the provided context, measuring factual consistency in RAG use cases. Scores range from 1 to 3, with higher values indicating stronger grounding in the context.	No
`Pointwise RAG Context Relevance`	Evaluates how relevant the retrieved context is to the user prompt, indicating the quality of the context retrieval stage in RAG systems. Scores range from 1 to 3, with higher values indicating that the context directly addresses the user prompt.	No
`Pointwise RAG Context Precision*`	Measures how much of the provided context is relevant and useful relative to the user prompt. Scores range from 1 to 3, with higher values indicating a greater proportion of relevant context.	No
`Pointwise RAG Completeness*`	Evaluates how effectively the model's response incorporates all relevant information from the provided context that is necessary to fully address the user prompt. Scores range from 1 to 5, with higher values indicating minimal relevant omissions.	No

Entries marked with an asterisk (*) are experimental metrics.

Related Information

Evaluations [page 654]

1.5.4.1.1 Language Match

Context

This metric determines whether LLM generated content is in the language required.

For more information about supported languages and their corresponding codes, see Supported Languages [page 689].

The following external libraries are used:

- lingua-py (Apache-2.0 License). For more information, see Lingua-py
- hanzidentifier (MIT) Hanzidentifier

Language match uses the following evaluation flow:

1. Use lingua-py to check the language
2. If lang_code not Chinese (simplified/traditional), match with user provided language code
3. Else run with hanzidentifier (zh_TW/zh_CN) and match with user provided language code

Procedure

Provide the code for the language you want to match.

- Provide an entry on a single row that will be applied to all rows.
- Provide an entry for each row individually.

For more information on available languages and their corresponding codes, see Supported Languages [page 689].

The metric returns `true` or `false` to indicate LLM output matches the given language. For example:

Sample Code

```
{
    "text": "LLM response, preferably long text",
    "language": "en"
}
```

Output Code

```
True
```

Supported Languages

Language	iso_code_639_1
AFRIKAANS	'AF'
ALBANIAN	'SQ'
ARABIC	'AR'
ARMENIAN	'HY'
AZERBAIJANI	'AZ'
BASQUE	'EU'
BELARUSIAN	'BE'
BENGALI	'BN'
BOKMAL	'NB'
BULGARIAN	'BG'
CATALAN	'CA'
CHINESE	'ZH'
SIMPLIFIED CHINESE	'ZH_CN'
TRADITIONAL CHINESE	'ZH_TW'
CROATIAN	'HR'
CZECH	'CS'
DANISH	'DA'
DUTCH	'NL'
ENGLISH	'EN'
ESPERANTO	'EO'
ESTONIAN	'ET'
FINNISH	'FI'
FRENCH	'FR'
GANDA	'LG'
GEORGIAN	'KA'
GERMAN	'DE'
GREEK	'EL'
GUJARATI	'GU'
HEBREW	'HE'
HINDI	'HI'
HUNGARIAN	'HU'
ICELANDIC	'IS'
INDONESIAN	'ID'

Language	iso_code_639_1
IRISH	'GA'
ITALIAN	'IT'
JAPANESE	'JA'
KAZAKH	'KK'
KOREAN	'KO'
LATIN	'LA'
LATVIAN	'LV'
LITHUANIAN	'LT'
MACEDONIAN	'MK'
MAORI	'MI'
MARATHI	'MR'
MONGOLIAN	'MN'
NYNORSK	'NN'
PERSIAN	'FA'
POLISH	'PL'
PORTUGUESE	'PT'
PUNJABI	'PA'
ROMANIAN	'RO'
RUSSIAN	'RU'
SERBIAN	'SR'
SHONA	'SN'
SLOVAK	'SK'
SLOVENE	'SL'
SOMALI	'SO'
SOTHO	'ST'
SPANISH	'ES'
SWAHILI	'SW'
SWEDISH	'SV'
TAGALOG	'TL'
TAMIL	'TA'
TELUGU	'TE'
THAI	'TH'
TSONGA	'TS'
TSWANA	'TN'

Language	iso_code_639_1
TURKISH	'TR'
UKRAINIAN	'UK'
URDU	'UR'
VIETNAMESE	'VI'
WELSH	'CY'
XHOSA	'XH'
YORUBA	'YO'
ZULU	'ZU'

1.5.4.1.2 JSON Schema Match

The `json_schema_match` metric validates whether the LLM's generated output conforms to a specified JSON schema. This is essential for applications that depend on structured data, requiring reliability and consistency for downstream tasks like API calls or database entries.

Dataset formatting

This metric requires a column or mapped variable "json_schema" in the dataset. This column must contain a valid JSON Schema object that defines the expected structure of the LLM's output.

A JSON Schema is a powerful tool for validating the structure of JSON data. For more information on how to construct a schema, see the official JSON Schema documentation https://json-schema.org/ .

The json_schema column entries can be provided in the following ways:

- Provide a schema for each row individually.
- Provide a schema in a single row that will be automatically applied to all rows.

The following example datasets show identical schemas provided in different ways:

Example 1: A single row is used to define all rows

Prompt	json_schema
Extract info from: 'User is Jane Doe, age 29.'	{"type": "object", "properties": {"name": {"type": "string"}, "age": {"type": "integer"}}, "required": ["name", "age"]}
Extract info from: 'User is John Smith, age 42.'	
Extract info from: 'User is Sam, age 35.'	

Example 2: Each row is defined individually

Prompt	json_schema
Extract info from: 'User is Jane Doe, age 29.'	{"type": "object", "properties": {"name": {"type": "string"}, "age": {"type": "integer"}}, "required": ["name", "age"]}
Extract info from: 'User is John Smith, age 42.'	{"type": "object", "properties": {"name": {"type": "string"}, "age": {"type": "integer"}}, "required": ["name", "age"]}
Extract info from: 'User is Sam, age 35.'	{"type": "object", "properties": {"name": {"type": "string"}, "age": {"type": "integer"}}, "required": ["name", "age"]}

Example Evaluation

This example uses the following JSON schema to assess LLM outputs for a task that extracts user contact and priority information. The schema is defined in the `json_schema` column of your dataset.

```
{
  "type": "object",
  "properties": {
    "customer_name": {
        "type": "string"
    },
    "customer_email": {
        "type": "string",
        "format": "email"
    },
    "priority": {
        "type": "string",
        "enum": ["High", "Medium", "Low"]
    }
  },
  "required": ["customer_name", "customer_email", "priority"]
}
```

Case 1: Valid Output (Pass)

This LLM output generates a well-structured JSON object that conforms perfectly to the schema:

Output Code

```
{
  "customer_name": "Alex",
  "customer_email": "alex@example.com",
  "priority": "High"
}
```

Metric Score: 1 (Pass).

The reasoning for this result is as follows:

- All required fields, for example `customer_name`, `customer_email` and `priority`, are present
- Their data types are correct. For example, `priority` is a string, and `High` is a valid enum option.

Case 2: Invalid Output (Fail)

This LLM output generates a JSON object that violates the schema in several ways:

Output Code

```
{
  "name": "Alex",
  "priority": 1
}
```

Metric Score: 0 (Fail).

The reasoning for this result is as follows:

- Missing Required Field: the required field `customer_email` is missing.
- Incorrect Field Name: the required field `customer_name` is missing; the LLM used name instead.
- Incorrect Data Type: the value for priority is the integer `1`, but the schema requires a string.
- Invalid Enum Value: the value 1 is not one of the allowed enum values: [`High`, `Medium`, `Low`].

1.5.4.1.3 Grounding Metrics

To use grounding related metrics, at least one of your runs provided should have `grounding_module_config` field present in your run configuration file. For more information, see Grounding [page 392].

Grounding integrates external, contextually relevant, domain-specific, or real-time data into AI processes. User provided context documents enhance the natural language processing capabilities of pretrained models, which are trained on general material.

This process involves an information retrieval component, where contextually relevant information is identified and retrieved and a generative component, where contextually relevant information is rewritten in line with the user query.

Evaluating a grounding system involves assessing both the retrieval and generation components to ensure they work together effectively.

You can evaluate your grounding pipelines using the metrics in this section.

1.5.4.1.3.1 Validated Metrics

Following are llm-as-a-judge metrics, which use another LLM to assess the response for specific criteria.

Groundedness

Groundedness measures whether the response relies solely on the provided context documents and avoids introducing external information or unsupported claims. A response is considered fully grounded if all factual claims are verifiable against the provided context.

Groundedness depends on the user query, context, and LLM output.

> **Note**
>
> While the definition of groundedness only depends on context and LLM output, we found that giving the user query as additional background information increased the correlation score with the ground truth from 67% to 79%.

Context Relevance

Context relevance measures how well the retrieved information from context documents matches the query.

Context relevance depends on the user query and context.

Validation Methodology

The above metrics were validated on the following datasets:

- Ragbench
- Two SAP internal datasets

Most datasets have ground truth scores, which were used to measure whether or not the metrics accurately captured performance. Ground truth scores are calculated based on a model's assessment of the ground truth data, and are verified by humans.

For datasets without ground truth scores, `gemini-2.5-pro` was used to generate synthetic scores, which were then verified by humans.

Tests for groundedness and context relevance using GPT-4o as a judge achieved a correlation of 79% and 83% with the ground truth scores, respectively.

Tests for groundedness and context relevance using GPT-4.1 as a judge achieved a correlation of 83.74% and 87.07% with the ground truth scores, respectively.

1.5.4.1.3.2 Computed Metrics vs LLM-as-a-Judge Metrics

Computed metrics such as BERT Score, BLEU, and cosine similarity performed poorly compared to llm-as-a-judge metrics and were not selected for grounding evaluation.

Instead, we recommend these corresponding metrics:

Computed Metric	LLM-as-a-Judge Metric
BERT Score with `user query` and `context`	Pointwise RAG Context Relevance
BERT Score with `context` and `llm output`	Pointwise RAG Groundedness

1.5.4.2 Custom Metrics

In addition to system-defined evaluation metrics, you can define custom metrics to evaluate a specific use case.

LLM-as-a-judge-type metrics are supported. A custom LLM-as-a-judge metric uses a judge LLM with a rubric to compute a metric rating. The output can be in numeric or text format.

You can define custom LLM-as-a-judge metrics using structured prompts.

Structured Prompts

Structured prompts are the recommended way to define custom metrics. They use a predefined format that you can use to define the metrics.

Sample Code

```
{
    "spec": {
        "promptType": "structured",
        "configuration": {
            "modelConfiguration": {
                "name": "<model name>",
                "version": "<model version>",
                #optional
                "parameters": [
                    {
                        "key": "temperature",
                        "value": 1.0
                    }
                ]
            },
            "promptConfiguration": {
                "definition": "Measures how well the response aligns with
verifiable facts",
                "evaluationTask": "Rate the factual accuracy of the response
compared to the reference",
                "ratingRubric": [
                    {
                        "rating": 5,
                        "rule": "Response is completely factual with no
unsupported claims"
                    },
                    {
                        "rating": 3,
                        "rule": "Response has minor inaccuracies but no major
contradictions"
                    },
                    {
                        "rating": 1,
                        "rule": "Response contains significant factual errors
or hallucinations"
                    }
                ],
                "criteria": "Evaluate based on facts, absence of
hallucinations, and alignment with reference",
                "evaluationSteps": [
                    "Compare response against reference material",
                    "Identify any factual claims in the response",
                    "Verify each claim against known facts",
```

```
                    "Rate based on accuracy level"
                ],
                "examples": [
                    {
                        "prompt": "What is the capital of France?",
                        "groundingInput": "Paris is the capital and most
populous city of France.",
                        "groundingOutput": "Paris is the capital and most
populous city of France.",
                        "response": "The capital of France is Paris.",
                        "reference": "Paris is the capital and most populous
city of France.",
                        "rating": 5,
                        "explanation": "Response is completely accurate and
matches the reference"
                    }
                ]
            }
        }
    }
}
```

1.5.4.2.1 Metric Management

You can manage the lifecycle of your custom metrics, outside of your evaluation deployments.

Perquisites

You're using the simplified workflow. For more information, see Custom Metrics [page 695]

You have prepared your custom metric for prompt management. For more information, see Register Evaluation Artifacts [page 656], and Create an Evaluation [page 658].

Context

You can:

- Define and create metrics in a structured way.
- Manage multiple versions of each metric definition as your use cases evolve.
- Track immutable revisions within each version, ensuring full reproducibility and auditability.

By separating metric definitions from your application logic, the service enables safe experimentation and updates without disrupting ongoing workflows. This ensures that new ideas can be tested, validated, and rolled out confidently while preserving the integrity of existing metrics.

1.5.4.2.1.1 Create a Custom Metric

Procedure

Send a POST request to the endpoint `$AI_API_URL/v2/lm/evaluationMetrics`.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Include the following in your request body:

Parameter	Description
`scenario`	The evaluation scenario ID and description this metric belongs to
`name`	Metric identifier (1–120 characters, no forward slashes)
`version`	Version string (1–10 characters)
`evaluationMethod`	"llm-as-a-judge"
`spec`	Your custom metric specification. For more information, see Custom Metrics [page 695].
`description` (optional)	Your choice of description of the metric (up to 512 characters)
`metricType` (optional) (deprecated) → Tip `metricType` is deprecated, therefore we recommend using `usageType` instead.	Purpose of the metric. Choose from: `agentic`, `optimization`, or `evaluation`. ⚠ Restriction For prompt optimization tasks: Only custom metrics that use the LLM-as-a-judge evaluation method and have numerical or Boolean output types are supported.

Parameter	Description
`usageType`	A list containing the purpose of the metric. The list can be empty, or contain one or more of the following: • `optimization` • `evaluation` **⚠ Restriction** For prompt optimization tasks: Only custom metrics that use the LLM-as-a-judge evaluation method and have numerical or Boolean output types are supported.
`includeProperties` (optional)	An array of context properties to include. Choose from: `prompt`, `grounding_query`, `grounding_response`, or `reference`.

⚠ Restriction

The maximum size of the payload of a custom metric object is 512 KB.

For tenants, the total number of custom metric objects including revisions is 50.

```
curl --request POST \
  --url $AI_API_URL/v2/lm/evaluationMetrics \
  --header 'AI-Resource-Group: $RESOURCE_GROUP' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer $TOKEN'
  --data '{
      "scenario": "qa-evaluation",
      "name": "factual-accuracy",
      "version": "1.0",
      "description": "Measures factual correctness of responses",
      "evaluationMethod": "llm-as-a-judge",
      "usageType": ["evaluation"],
      "includeProperties": ["prompt", "reference"],
      "spec": ...  // Custom Metric Definition
}'
```

Results

Your response includes one of the following response codes:

- **Successful** response codes:
  - **201**
- **Error** response codes:
  - **400 Bad Request**: Invalid request format, missing required fields, or validation errors
  - **403 Forbidden**: Insufficient permissions or invalid tenant authorization
  - **409 Conflict**: A metric with the same scenario, name, and version already exists

Successful responses include a unique ID that can be used to retrieve your custom metric later.

Output Code

```
{
"message": "<success message>"
"id": "<metric ID>"
"scenario": "<scenario ID>"
"name": "<name>"
"version": "<version>"
}
```

1.5.4.2.1.2 Retrieve Custom Metrics

You can use the following endpoints to view your custom metrics and their details:

Listing Custom Metrics

You can retrieve a list of your custom metrics by sending a GET request to the endpoint `$AI_API_URL/v2/lm/evaluationMetrics`.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

```
curl --request GET \
  --url $AI_API_URL/v2/lm/evaluationMetrics \
  --header 'AI-Resource-Group: {{resource_group}}' \
  --header 'Authorization: Bearer {{access_token}}'
```

This endpoint returns a response in JSON format containing details of your custom metrics.

You can add filters to your list by including the following parameters:

- `scenario`: Filter metrics by scenario ID
- `name`: Filter metrics by metric name (for example, "groundedness")
- `version`: Filter metrics by specific version
- `evaluationMethod`: Filter by evaluator type (for example., "llm-as-a-judge")
- `retrieve`: What type of metrics to retrieve - "imperative", "declarative", or "both" (default: "both")
- `includeSpec`: Whether to include the full metric specification in response (default: false)
- `includeSystemMetrics`: Whether to include system-defined metrics (default: true)

For example:

```
curl --request GET \
```

```
  ---url $AI_API_URL/v2/lm/evaluationMetrics \
  --header 'AI-Resource-Group: $RESOURCE_GROUP' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer $TOKEN'
  --data '{
    "params" = {
        "scenario": "genai-evaluations",
        "includeSpec": "true"
        "name": "factual-accuracy",
        "evaluationMethod": "llm-as-a-judge"
    }
}'
```

Error responses include:

- **400 Bad Request**: Invalid query parameters or request format
- **403 Forbidden**: Insufficient permissions or invalid tenant authorization

Succesful responses include the following:

Output Code

```
{
"count": ... ,  // Number of items retrieved
"resources": ... , // List of MetricItem
}
```

Get a Metric by ID

You can retrieve the details of a single metric using its ID by sending a GET request to the endpoint `$AI_API_URL/v2/lm/evaluationMetrics/$EVALUATION_METRIC_ID`.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Populate the code snippet with the following:

- Your metric ID

```
curl --request GET \
  --url $AI_API_URL/v2/lm/evaluationMetrics/$EVALUATION_METRIC_ID \
  --header 'AI-Resource-Group: $RESOURCE_GROUP' \
  --header 'Authorization: Bearer $TOKEN'
```

This endpoint returns a response in JSON format containing details of your custom metric.

Error responses include:

- **400 Bad Request**: Invalid UUID (metric ID) format

- **403 Forbidden**: Insufficient permissions or invalid tenant authorization
- **404 Not Found**: Metric with the specified UUID (metric ID) does not exist

Metrics are returned in the following format:

Output Code

```
{"scenario": ... ,
"name": ... ,
"version": ... ,
"evaluationMethod": ... ,
"id": ... ,
"spec": ... ,
"description": ... ,
"usageType": ... ,
"includeProperties": ... ,
"managedBy": ... ,
"createdAt": ... ,
"additionalProperties": ...
}
```

Get Metric History

You can retrieve the history of a single metric by sending a GET request to the endpoint `$AI_API_URL/v2/lm/scenarios/$SCENARIO/evaluationMetrics/$NAME/versions/$VERSION/history.`.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Populate the code snippet with the following:

- `scenario`: The scenario the metric belongs to
- `name`: The metric name
- `version`: The specific version of the metric

```
curl --request GET \
--url $AI_API_URL/v2/lm/scenarios/$SCENARIO/evaluationMetrics/$NAME/versions/
$VERSION/history. \
  --header 'AI-Resource-Group: $RESOURCE_GROUP' \
  --header 'Authorization: Bearer $TOKEN'
```

This endpoint returns a response in JSON format containing a count and an array of custom metrics of the specified scenario, name, and version, sorted from newest to oldest.

Error responses include:

- **400 Bad Request**: Invalid path parameters or request format

- **403 Forbidden**: Insufficient permissions or invalid tenant authorization
- **404 Not Found**: Metric with the specified scenario/name/version does not exist

Output Code

```
{
"count": ... ,  // Number of items retrieved
"resources": ... , // List of MetricItem
}
```

1.5.4.2.1.3 Delete a Custom Metric

You can delete a single metric using its ID by sending a DELETE request to the endpoint `$AI_API_URL/v2/lm/evaluationMetrics/$EVALUATION_METRIC_ID`.

Caution

Deletion cannot be reversed.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Populate the code snippet with the following:

- Your metric ID

This endpoint returns a response code. Successful requests return code 204.

```
curl --request DELETE \
  --url {{AI_API_URL}}/v2/lm/evaluationMetrics/{{evaluationMetricId}} \
  --header 'AI-Resource-Group: $RESOURCE_GROUP' \
  --header 'Authorization: Bearer $TOKEN'
```

Error responses include:

- **403 Forbidden**: Insufficient permissions or invalid tenant authorization
- **404 Not Found**: Metric with the specified UUID (metric ID) does not exist

1.6 Additional Concepts

1.6.1 Create a Deployment for Orchestration

Prerequisites

> **Note**
>
> It is likely that you already have an orchestration deployment running in your default resource group. These steps are only required for users who do not have an orchestration deployment or who would like to use model restriction. For more information, see Get An Orchestration Deployment URL [page 7] or Model Restriction [page 592].

- You have an SAP AI Core service instance and service key. For more information, see SAP AI Core Initial Setup Documentation.
- You're using the `extended` service plan. For more information, see Service Plans and Update a Service Plan.
- You have completed the client authorization for your preferred user interface. For more information, see Use a Service Key.

Context

You create a deployment to make orchestration capabilities available for use. After the deployment is complete, you get a `deploymentUrl`. You can use this URL across your organization to access orchestration in the generative AI hub.

For more information, see Consumption of GenAI Models Using Orchestration – A Beginner's Guide as well as Libraries and SDKs.

Procedure

1. Create a configuration for the orchestration deployment.

   Orchestration is exposed via the global scenario `orchestration` and the executable `orchestration`.

   a. Send the following request:

```
curl --request POST "$AI_API_URL/v2/lm/configurations" \
  --header "Authorization: Bearer $TOKEN" \
  --header "AI-Resource-Group: $RESOURCE_GROUP" \
  --header "Content-Type: application/json" \
  --data-raw  '{
```

```
    "name": "orchestration-config",
    "executableId": "orchestration",
    "scenarioId": "orchestration"
  }'
```

The response appears as follows:

Output Code

```
{
  "id": "f7ac7f77-e70f-4e9c-86b5-1504b44fe789",
  "message": "Configuration created"
}
```

b. Take the `id` of the configuration and set it as an environment variable.

```
ORCH_CONFIG_ID=f7ac7f77-e70f-4e9c-86b5-1504b44fe789
```

2. Create the deployment for orchestration using the configuration `id`:

a. Send the following request:

```
curl --request POST $AI_API_URL/v2/lm/deployments \
    --header 'content-type: application/json' \
    --header "Authorization: Bearer $TOKEN" \
    --header "AI-Resource-Group: $RESOURCE_GROUP" \
    --data-raw "{
        \"configurationId\": \"$ORCH_CONFIG_ID\"
    }"
```

The response appears as follows:

Output Code

```
{
  "deploymentUrl": "",
  "id": "d4168482710c6cf9",
  "message": "Deployment scheduled.",
  "status": "UNKNOWN"
}
```

b. Take the `id` of the deployment response and set it as an environment variable.

```
ORCH_DEPLOYMENT_ID=d4168482710c6cf9
```

3. Wait for the deployment to start.

It takes a few minutes for the orchestration deployment to reach the status RUNNING. You can check the status of the deployment using the deployment `id`.

```
curl --request GET "$AI_API_URL/v2/lm/deployments/$ORCH_DEPLOYMENT_ID" \
    --header "Authorization: Bearer $TOKEN" \
    --header "AI-Resource-Group: $RESOURCE_GROUP"
```

If the output looks as follows (that is, the status is RUNNING), the deployment is ready. If not, try again after a couple of minutes.

```
{
  "configurationId": "f7ac7f77-e70f-4e9c-86b5-1504b44fe789",
  "configurationName": "orchestration-config",
  "createdAt": "2024-06-20T14:40:16Z",
```

```
  "deploymentUrl": ,
  "details": {
    "resources": {
      "backend_details": {}
    },
    "scaling": {
      "backend_details": {}
    }
  },
  "id": "d4468482710c6cf9",
  "lastOperation": "CREATE",
  "latestRunningConfigurationId": "f7ac7f77-e70f-4e9c-86b5-1504b44fe789",
  "modifiedAt": "2024-06-20T14:48:03Z",
  "scenarioId": "llm-orchestration",
  "startTime": "2024-06-20T14:41:22Z",
  "status": "RUNNING",
  "submissionTime": "2024-06-20T14:40:18Z",
  "targetStatus": "RUNNING"
}
```

4. Set the value of the returned `deploymentUrl` as an environment variable.

   The `deploymentUrl` is available only when the deployment has the status RUNNING.

   ```
   ORCH_DEPLOYMENT_URL="<deployment_url>"
   ```

1.6.2 Deployment Lifecycle Management

Context

> ⚠ Restriction
>
> Only deployments in a stopped state can be deleted.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Stop a Deployment

Send a PATCH request to the endpoint `$AI_API_URL/v2/lm/deployments/<deploymentid>`. Include the ID of the deployment that you want to stop in your request:

Sample Code

```
curl PATCH '$AI_API_URL/v2/lm/deployments/<deploymentid>' \
--header 'AI-Resource-Group: default' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
"targetStatus": "STOPPED"
}'
```

Delete a Deployment

Send a DELETE request to the endpoint `$AI_API_URL/v2/lm/deployments/<deploymentid>`. Include the ID of the deployment that you want to stop in your request.

Sample Code

```
curl DELETE '$AI_API_URL/v2/lm/deployments/<deploymentid>' \
--header 'AI-Resource-Group: default' \
--header "Authorization: Bearer $AUTH_TOKEN" \
```

1.6.3 Administration

1.6.3.1 Manage Your Git Repository

1.6.3.1.1 Add a Git Repository

You can use your own git repository to version control your SAP AI Core templates.

Using the API

Prerequisites

- You've completed the initial setup.
- You have access to a git repository over the Internet.

- You've generated a personal access token for your git repository. For more information, see Create a Personal Access Token .
- If you want to onboard a git repository hosted on GitLab, make sure that the repository URL contains the `.git` suffix.
- Secrets aren't permitted in your repository. If secrets are used, it isn't possible to synchronize content.

> **Note**
>
> When you synchronize resources, make sure that there are no naming collisions, especially if you use multiple repositories or applications in one tenant. If you experience difficulties during synchronization, we recommend that you use only one repository or application per tenant.
>
> For example, the following repository URLs are all considered the same repository:
>
> - `https://github.com/user/repo`
> - `https://github.com/user/repo/`
> - `https://github.com/user/REPO/`

Context

Git repositories are managed by creating personal access tokens and adding them in SAP AI Core. Personal access tokens are a means of allowing and controlling connections to GitHub repositories without compromising your credentials.

> **→ Remember**
>
> You are responsible for the rotation of your access credentials and certificates of SAP AI Core within BTP according to regional policy.

Procedure

Submit a POST request to the endpoint `$AI_API_URL/v2/admin/repositories` and include your credentials in your request.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $AUTH_TOKEN
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Populate your request with the following:

- Repository name (optional)

- Your credentials

```
curl --location --request POST "$AI_API_URL/v2/admin/repositories" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--data-raw '{
    "name": "<yourChoiceOfRepositoryName>",
    "url": "https://github.com/john/examplerepo",
    "username": "john",
    "password": "<GIT_PAT_USER_TOKEN>"
}'
```

You specify your unique git repository details as follows:

- `url`: URL of the git repository

  > ⚠ Restriction
  >
  > - Only ASCII alphanumerics, digits, and the characters ".", " -", "_" and "%" are allowed.
  > - Do not use IP addresses as repository URLs.

- `username`: (Service) user that's accessing the git repository
- `password`: git personal access token. For more information, see Create a Personal Access Token .

> → Tip
>
> To share a repository between two tenants, add the repository in SAP AI Core separately for each tenant and provide the **same** `username` and `password`.

Results

Successful responses return code **200** and include a success message and ID.

> {·} Output Code

```
{
  "id": "<yourRepositoryId>",
  "message": "The repository has been on-boarded"
}
```

Related Information

Initial Setup

1.6.3.1.2 Edit a Git Repository

You can edit a Git repository using its name to update your authentication details. This lets you maintain secure access and keep workflows synced with the latest code.

Using the API

Procedure

Submit a PATCH request to the endpoint `$AI_API_URL/v2/admin/repositories/$REPOSITORY_NAME` and include your changes.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $AUTH_TOKEN
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Populate your request with the following:

- Repository name
- Updates to your credentials

```
curl --location --request PATCH "$AI_API_URL/v2/admin/repositories/
$REPOSITORY_NAME" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--data-raw '{
        "username": "john",
        "password": "<GIT_PAT_USER_TOKEN>"
    }'
```

Results

Successful responses return code **200** and include a success message and ID.

Output Code

```
{
  "id": "<yourRepositoryId>",
  "message": "The repository credentials have been updated and will
eventually be synced"
}
```

1.6.3.1.3 Delete a Git Repository

You remove a Git repository from a connection if its URL is invalid or contains errors, or if the repo is no longer required. Once a Git repository is removed, it can no longer be selected as a source repository for an application.

Using a Third-Party API Platform

Send a DELETE request to the endpoint `{{apiurl}}/v2/admin/repositories/{{repositoryName}}` and include your repository name.

Using curl

```
curl --location --request DELETE "{{apiurl}}/v2/admin/repositories/
{{repositoryName}}" \
```

Using Curl

Context

You remove a Git repository from a connection if its URL is invalid or contains errors, or if the repository is no longer required. Once a Git repository is removed, it can no longer be selected as a source repository for an application.

Procedure

Run the following code:

```
curl --location --request DELETE "{{apiurl}}/v2/admin/repositories/
{{repositoryName}}" \
```

Using a Third-Party API Platform

Context

You remove a Git repository from a connection if its URL is invalid or contains errors, or if the repository is no longer required. Once a Git repository is removed, it can no longer be selected as a source repository for an application.

Procedure

Send a DELETE request to the endpoint `{{apiurl}}/v2/admin/repositories/{{repositoryName}}` and include your repository name.

1.6.3.2 Manage Resource Groups

A resource group is a unique dedicated namespace or workspace environment, where users can create or add configurations, executions, deployments, and artifacts. They are used for running training jobs or model servers.

Resource groups are used to physically isolate machine learning workloads, and to logically isolate related resources for a usage scenario.

When your tenant is onboarded, a default resource group is automatically created. Default resource groups can't be deleted.

As an administrator, you create, edit, or delete resource groups, based on your service consumers and usage scenarios.

Runtime entities such as executions, deployments, configurations, and artifacts belong to a specific resource group and aren't shared across resource groups. Scenarios, executables, and Docker registry secrets are shared by all resource groups within a tenant.

A resource group is also referred to as an instance.

> → Remember
>
> Your SAP global account can consist of several accounts. Each account can be associated with a tenant. A tenant can contain multiple resource groups. A tenant always contains a default resource group, as well as the resource groups defined for your usage scenarios.

> ⓘ Note
>
> The maximum number of resource groups is limited at tenant level to 50. If you reach this limit, you receive an error message. To free up space, delete some resource groups. Alternatively, raise a ticket to increase your quota.
>
> For more information, see Delete a Resource Group [page 714].

Edit a Resource Group [page 712]
: You edit a resource group to update its configuration, such as changing labels, adjusting resource limits, or modifying access settings.

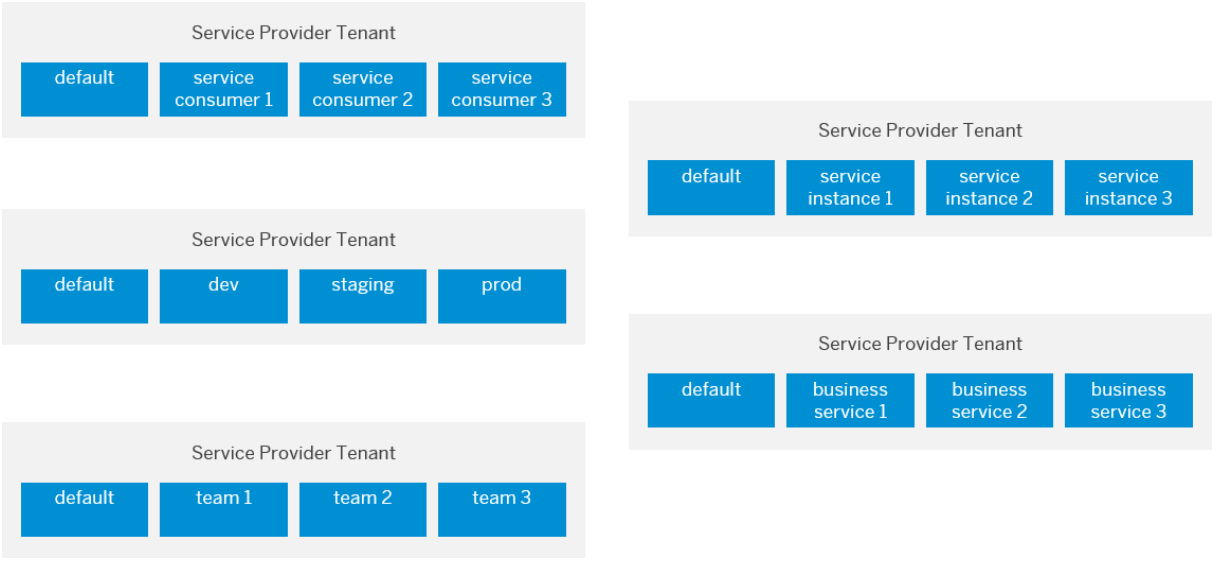
Delete a Resource Group [page 714]
: Deletes a resource group that is invalid, contains errors, or is no longer required.

Resource Group Level Resources

Executables at the tenant level are shared across all resource groups. In contrast, runtime entities such as executions, deployments, configurations, and artifacts belong to a specific resource group and cannot be shared across resource groups. Similarly, generic secrets created within a resource group can be used only for workloads within that group.

You can register an object store at the resource-group level by setting the resource group header. We recommend that you do not use the same object store bucket with the same IAM user for multiple resource groups.

Example resource group mappings are outlined in the figure below:



1.6.3.2.1 Edit a Resource Group

You edit a resource group to update its configuration, such as changing labels, adjusting resource limits, or modifying access settings.

Parent topic: Manage Resource Groups [page 711]

Related Information

Delete a Resource Group [page 714]

Using the API

Context

> **Note**
>
> Resource group Ids must be of length minimum: 3, maximum: 253. The first and last characters must be either a lowercase letter, an uppercase letter, or a number. Character entries from the second to penultimate can include a lower case letter, an upper case letter, a number, a period (.), or a hyphen (-). No other special characters are permitted.

Procedure

Edit a resource group by sending the following:

```
curl --location --request PATCH "$AI_API_URL/v2/admin/resourceGroups/
{{resource_group_name}}"
--header "Authorization: Bearer $TOKEN"
--header 'Content-Type: application/json'
--data-raw '{
"resourceGroupId": "<ID of your resource group>"
}'
```

Results

Successful responses return code **202** and include a success message.

Using a Third-Party API Platform

Context

> **Note**
>
> Resource group Ids must be of length minimum: 3, maximum: 253. The first and last characters must be either a lowercase letter, an uppercase letter, or a number. Character entries from the second to

penultimate can include a lower case letter, an upper case letter, a number, a period (.), or a hyphen (-). No other special characters are permitted.

Procedure

Send a PATCH request to the endpoint `{{apiurl}}/v2/admin/resourceGroups/{{resource_group_name}}` with the body:

```
{
    "resourceGroupId": "<ID of your resource group>"
}
```

1.6.3.2.2 Delete a Resource Group

Deletes a resource group that is invalid, contains errors, or is no longer required.

Parent topic: Manage Resource Groups [page 711]

Related Information

Edit a Resource Group [page 712]

Using the API

Context

→ Remember

When you delete a resource group, Manage mTLS Certificate Secrets [page 720] in that group are deleted. Certificates that were issued before the deletion may remain valid until they expire. You are responsible for removing or revoking trust on any external services that rely on those certificates.

Procedure

Run the following code:

```
curl --location --request POST "$AI_API_URL/v2/admin/resourceGroups/
{{resource_group_name}}"
--header "Authorization: Bearer $TOKEN"
--header 'Content-Type: application/json'
--data-raw '{
"resourceGroupId": "<ID of your resource group>"
}'
```

Results

Successful responses return code **202** and include a success message.

Using a Third-Party API Platform

Context

> → Remember
>
> When you delete a resource group, Manage mTLS Certificate Secrets [page 720] in that group are deleted. Certificates that were issued before the deletion may remain valid until they expire. You are responsible for removing or revoking trust on any external services that rely on those certificates.

Procedure

Send a DELETE request to the endpoint `{{apiurl}}/v2/admin/resourceGroups/{{resource_group_name}}`.

1.6.3.3 Manage Generic Secrets

1.6.3.3.1 Update a Generic Secret

To update a generic secret, use the PATCH endpoint as shown below. The PATCH operation updates the data and/or labels provided. This can be used for rotating secret credentials.

Using Curl

Context

Generic secrets store sensitive information when system secrets aren't applicable. They're useful in integration scenarios where SAP AI Core acts as an orchestration layer.

SAP AI Core lets you use generic secrets at various levels:

- Main-tenant scope
- Tenant-wide level
- Resource-group level

Generic secrets differ from system secrets, like those for object stores or Docker registries. They store sensitive information for the main tenant, all resource groups, or individual resource groups via an API. You can attach these secrets to containers in executions or deployments as environment variables or volume mounts.

To allow the rotation of tenant-wide secrets for long-running deployments without requiring a restart, the deployment must mount the tenant-wide secret. It must also monitor the mounted secret for changes instead of relying on an in-memory copy. When a tenant-wide secret is updated, the tenant must observe the `resourceGroupSecretReplicationStatus` field in the `Get Secret` endpoint to confirm that the secret has been successfully replicated across the required resource groups. For more information, see Consume Generic Secrets in Executions or Deployments.

Procedure

Submit a PATCH request to the endpoint `/v2/admin/secrets/"$SECRET_NAME"`. Specify the scope via the headers:

Specify the scope of the request via the header `AI-Tenant-Scope` and specify the scope via the or `AI-Resource-Group`:

- **`AI-Tenant-Scope`** : **`true`**. The operation will be performed at the main-tenant level.
- **`AI-Resource-Group`** : **`<resource-group-name>`**. The operation will be performed at the resource-group level.
- **`AI-Tenant-Scope`** : **`true`** and **`AI-Resource-Group`**: **`*`**. The operation will be performed at the tenant-wide level.

```
curl --location --request PATCH "$AI_API_URL/v2/admin/secrets/$SECRET_NAME" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: default' \
--data-raw '{
        "data": {
            "some-credential": "bXktc2Vuc2l0aXZlLWRhdGE="
            }
        "labels": [
            {"key": "ext.ai.sap.com/<key1>", "value": "<value1>"},
            {"key": "ext.ai.sap.com/<key2>", "value": "<value2>"}
            ]
}'
```

Results

Output Code

```
{
"message": "The secret has been modified",
"name": "my-generic-secret"
}
```

The response confirms the successful update of the secret. Label updates are applied immediately without requiring secret recreation. You can verify label updates by retrieving the secret using the GET endpoint.

Using a Third-Party API Platform

Context

Generic secrets store sensitive information when system secrets aren't applicable. They're useful in integration scenarios where SAP AI Core acts as an orchestration layer.

SAP AI Core lets you use generic secrets at various levels:

- Main-tenant scope
- Tenant-wide level
- Resource-group level

Generic secrets differ from system secrets, like those for object stores or Docker registries. They store sensitive information for the main tenant, all resource groups, or individual resource groups via an API. You can attach these secrets to containers in executions or deployments as environment variables or volume mounts.

To allow the rotation of tenant-wide secrets for long-running deployments without requiring a restart, the deployment must mount the tenant-wide secret. It must also monitor the mounted secret for changes instead of relying on an in-memory copy. When a tenant-wide secret is updated, the tenant must observe the `resourceGroupSecretReplicationStatus` field in the `Get Secret` endpoint to confirm that the secret has been successfully replicated across the required resource groups. For more information, see Consume Generic Secrets in Executions or Deployments.

Procedure

1. Send a PATCH request to the endpoint `{{apiurl}}/v2/admin/secrets/{{secretName}}`

   As the request body, select the *raw* radio button and enter the following code:

   Source Code

   ```
   {
       "data": {
           "updated-credentials": "bXktc2VjcmV0LW90aGVyLWNyZWRlbnRpYWw="
       }
   ```

```
    "labels": [
            {"key": "ext.ai.sap.com/<key1>", "value": "<value1>"},
            {"key": "ext.ai.sap.com/<key2>", "value": "<value2>"}
            ]
}
```

Specify the scope of the request via the header `AI-Tenant-Scope` and specify the scope via the or `AI-Resource-Group`:

- **`AI-Tenant-Scope`** : **`true`**. The operation will be performed at the main-tenant level.
- **`AI-Resource-Group`** : **`<resource-group-name>`**. The operation will be performed at the resource-group level.
- **`AI-Tenant-Scope`** : **`true`** and **`AI-Resource-Group`**: **`*`**. The operation will be performed at the tenant-wide level.

You can update labels alongside or instead of secret data. Only labels with the `ext.ai.sap.com/` prefix can be modified.

> ⚠ Restriction
>
> The following labels cannot be updated via PATCH:
> - `ext.ai.sap.com/document-grounding`
> - `ext.ai.sap.com/documentRepositoryType`

To remove a label, set its value to an empty string ("").

2. Send the request.

Results

> {·} Output Code

```
{
"message": "The secret has been modified",
"name": "my-generic-secret"
}
```

The response confirms the successful update of the secret. Label updates are applied immediately without requiring secret recreation. You can verify label updates by retrieving the secret using the GET endpoint.

1.6.3.3.2 Delete a Generic Secret

To get a secret name, see Get Generic Secrets.

Using Curl

Procedure

Submit a DELETE request to the endpoint `/v2/admin/secrets/"$SECRET_NAME"`. Specify the scope of the request via the header `AI-Tenant-Scope` and `AI-Resource-Group`:

- **`AI-Tenant-Scope`** : **`true`**. The operation will be performed at the main-tenant level.
- **`AI-Resource-Group`** : **`<resource-group-name>`**. The operation will be performed at the resource-group level.
- **`AI-Tenant-Scope`** : **`true`** and **`AI-Resource-Group`**: **`*`**. The operation will be performed at the tenant-wide level.

In this example we use the resource-group scope:

```
curl --location --request DELETE "$AI_API_URL/v2/admin/secrets/$SECRET_NAME" \
--header "Authorization: Bearer $TOKEN" \
--header 'AI-Resource-Group: default'
```

Using a Third-Party API Platform

Procedure

1. Send a DELETE request to the endpoint**`{{apiurl}}/v2/admin/secrets/{{secretName}}`**

   As the request body, select the *none* radio button. Specify the scope of the request via the header `AI-Tenant-Scope` or `AI-Resource-Group`:
   - **`AI-Tenant-Scope`** : **`true`**. The operation will be performed at the main-tenant level.
   - **`AI-Resource-Group`** : **`<resource-group-name>`**. The operation will be performed at the resource-group level.
   - **`AI-Tenant-Scope`** : **`true`** and **`AI-Resource-Group`**: **`*`**. The operation will be performed at the tenant-wide level.
2. Send the request.

Results

Output Code

```
200
```

1.6.4 Manage mTLS Certificate Secrets

Context

mTLS certificate secrets let your SAP AI Core workloads authenticate to external services using mutual TLS (mTLS). SAP AI Core automatically retrieves X.509 client certificates from the SAP BTP Certificate Service on your behalf, eliminating the need for manual certificate management. To establish mTLS connections, you mount this certificate in your executions or deployments.

> ⚠ Restriction
>
> mTLS certificate secrets are managed via the API and are available through SAP AI Core only.

When you create an mTLS certificate secret, the following procedure occurs:

1. SAP AI Core requests a client certificate from the SAP BTP Certificate Service.
2. The certificate and private key are stored as a TLS secret in your resource group's namespace.
3. The response returns the `subjectDN` (subject Distinguished Name) of the issued certificate.
4. You configure trust for this subject DN on your external target service.
5. Your workload mounts the secret as a volume and uses the certificate files for mTLS authentication.

The certificate's subject DN includes organizational fields set by the BTP Certificate Service (such as `C=`, `O=`, `OU=`) and a `CN=` (common name) field that you can customize.

> ⓘ Note
>
> The certificate and private key content are never returned by the API. They're only accessible inside a workload container through a volume mount. The API returns only certificate metadata, such as subject DN, expiry, serial number, and common name.

Establishing Trust on the Target Service

After creating the secret, you must configure the external target service to trust certificates matching the subject DN returned in the creation response.

> ⓘ Note
>
> The exact steps depend on the target service.

To establish trust on the target service, you need to achieve the following:

- Register the SAP Cloud Root CA (or the specific intermediate CA) as a trusted issuer.
- Configure an access policy or trust rule that maps the certificate's subject DN pattern to the appropriate authorization level.

→ Remember

If you delete an mTLS certificate secret or stop using it, you should also remove the trust configuration from the external service. Certificates may remain valid until their original expiry date even after the secret is deleted. For more information, see Delete an mTLS certificate secret [page 730] and Service Offboarding

Certificate Lifecycle and Rotation

mTLS certificates have a limited validity period. You can check the `expiresAt` field via the `GET` endpoint to see when the certificate expires. For more information, see List mTLS Certificate Secrets [page 726].

⚠ Caution

You're responsible for monitoring certificate expiry and rotating secrets before they expire.

To rotate a certificate, send a `PATCH` request to the secret's endpoint. Rotation generates a new certificate and key with the same subject DN. The old certificate remains valid until its original expiry, ensuring a seamless transition with no downtime. For more information, see Rotate an mTLS Certificate Secret [page 728].

Recommended Rotation Practices

The following best practices are recommended:

1. Monitor the `expiresAt` field of your mTLS certificate secrets regularly.
2. Rotate the secret well before the expiry date (for example, 30 days in advance).
3. If your workload caches the certificate in memory: restart it after rotation to pick up the new certificate. If your workload reads the certificate from disk on each request: the updated files are propagated automatically.

Scope and Limitations

- mTLS certificate secrets are scoped to a single resource group. They don't support main-tenant scope, tenant-wide scope, or shared resource groups.
- Each secret produces one certificate. If you need multiple certificates (for example, for different external services), create multiple secrets.
- The `commonName` field is optional and limited to 64 characters. If omitted, the secret name is used.
- Secret names must consist of lowercase alphanumeric characters, hyphens, or dots only, with a maximum length of 252 characters.

Related Information

Create an mTLS Certificate Secret [page 722]
List mTLS Certificate Secrets [page 726]
Rotate an mTLS Certificate Secret [page 728]

Consume mTLS Certificate Secrets [page 724]
Get Details of an mTLS Certificate Secret [page 727]
Delete an mTLS certificate secret [page 730]

1.6.4.1 Create an mTLS Certificate Secret

Create an mTLS certificate secret so that your workloads can make mutually authenticated TLS (mTLS) requests to external services.

Prerequisites

Make sure that you completed the initial setup. For more information, see Initial Setup.

Context

An `mTLS` certificate secret instructs SAP AI Core to obtain an X.509 client certificate from the SAP BTP Certificate Service on your behalf. Unlike generic secrets, where you provide the credential payload, SAP AI Core generates both the certificate and the private key.

You can mount the generated certificate in executions or deployments to enable mTLS-authenticated requests to external services that trust the certificate's subject DN.

`mTLS` certificate secrets are scoped to a single resource group. They do not support main-tenant scope, tenant-wide scope, or shared resource groups.

When you create the secret, the response includes the `subjectDN`. You must configure trust for this subject DN on the external service before authentication succeeds.

You can optionally specify a `commonName`. If you omit it, SAP AI Core uses the secret name as the common name. The common name appears in the Common Name field of the certificate's subject distinguished name.

> **Note**
>
> Creation is asynchronous. After you send a successful `POST` request (HTTP 202), the certificate may take a few moments to become available. Use a `GET` request to verify readiness before you run an execution or deployment that mounts the secret.

Procedure

1. Send a POST request to the endpoint `$AI_API_URL/v2/admin/mtlsCertificateSecrets` .
2. Include the `AI-Resource-Group` header to specify the resource group.

Minimal request (common name defaults to secret name):

```
curl --location --request POST "$AI_API_URL/v2/admin/mtlsCertificateSecrets" \
--header "Authorization: Bearer $TOKEN" \
--header "Content-Type: application/json" \
--header "AI-Resource-Group: default" \
--data-raw '{
    "name": "my-mtls-cert"
}'
```

Request with a custom common name:

```
curl --location --request POST "$AI_API_URL/v2/admin/mtlsCertificateSecrets" \
--header "Authorization: Bearer $TOKEN" \
--header "Content-Type: application/json" \
--header "AI-Resource-Group: default" \
--data-raw '{
    "name": "my-mtls-cert",
    "data": {
        "commonName": "my-service-client"
    }
}'
```

> **Note**
>
> The secret name must match the pattern `^[a-z0-9\-\.]+$` and be at most 252 characters. A common name can be up to 64 characters.

Results

The service returns HTTP `202 Accepted` and includes the issued certificate's subject DN.

Output Code

```
202 Accepted
{
    "name": "my-mtls-cert",
    "message": "...",
    "subjectDN": "C=DE, O=SAP SE, OU=SAP Cloud Platform Clients,
OU=<subaccount-ou>, L=<locality-hash>, CN=my-service-client"
}
```

Use the returned `subjectDN` to configure trust on the external service.

Using a Third-Party API Platform

Procedure

1. Send a POST request to `{{apiurl}}/v2/admin/mtlsCertificateSecrets.`

2. Select *raw* for the request body and enter the JSON payload.

```
{
    "name": "my-mtls-cert",
    "data": {
        "commonName": "my-service-client"
    }
}
```

3. Set the `AI-Resource-Group` header to the target resource group name.
4. Send the request.

1.6.4.2 Consume mTLS Certificate Secrets

You can consume mTLS certificate secrets in your workloads to authenticate securely with external services that require mutual TLS.

The `mTLS` certificate secrets can be mounted as volumes in your execution or deployment containers. The secret provides a certificate and private key in PEM format. Your workload uses these files to send mTLS-authenticated requests to external services.

> **Note**
>
> `mTLS` certificate secrets can only be consumed as volume mounts. You can't use them as environment variables.

Mount an mTLS Certificate Secret as a Volume

To mount the certificate and key files in your workload container, define a volume referencing the secret and a corresponding volume mount in your workflow or serving template.

The secret contains two data keys:

- `tls.crt` — the PEM-encoded X.509 client certificate
- `tls.key` — the PEM-encoded private key

You can project these keys to custom file paths by using the `items` field.

Example workflow template:

```
apiVersion: argoproj.io/v1alpha1
kind: WorkflowTemplate
metadata:
  name: my-mtls-workflow
  annotations:
    scenarios.ai.sap.com/name: "my-scenario"
    executables.ai.sap.com/name: "my-executable"
spec:
  entrypoint: main
  templates:
    - name: main
      inputs:
```

```
        parameters:
          - name: mtlsSecretName
      volumes:
        - name: mtls-cert-volume
          secret:
            secretName: "{{inputs.parameters.mtlsSecretName}}"
            items:
              - key: tls.crt
                path: client.crt
              - key: tls.key
                path: client.key
      container:
        image: my-image:latest
        volumeMounts:
          - name: mtls-cert-volume
            mountPath: /mnt/mtls
            readOnly: true
```

Inside the container, the following files are available:

- `/mnt/mtls/client.crt` — the client certificate
- `/mnt/mtls/client.key` — the private key

Your application can use these files to configure an mTLS-authenticated HTTP client. For example, in Python:

```
import requests
response = requests.get(
    "https://external-service.example.com/api/data",
    cert=("/mnt/mtls/client.crt", "/mnt/mtls/client.key")
)
```

Parameterize the Secret Name

You can parameterize the secret name so that it's provided during execution creation instead of being hardcoded. Use the `inputs.parameters` mechanism:

```
inputs:
  parameters:
    - name: mtlsSecretName
```

When creating the execution through the AI API, provide the parameter value:

```
{
    "parameterBindings": [
        {
            "key": "mtlsSecretName",
            "value": "my-mtls-cert"
        }
    ]
}
```

Additional Information

- Only `mTLS` certificate secrets that belong to the same resource group as the execution or deployment can be mounted. The platform validates this during admission.
- If the referenced secret doesn't exist or doesn't belong to the same resource group, the workload is rejected.
- After certificate rotation, see Rotate an mTLS Certificate Secret [page 728], the mounted files update automatically. If your application caches the certificate in memory, you must restart the workload to use the new certificate.

1.6.4.3 List mTLS Certificate Secrets

Retrieve a list of mTLS certificate secrets in a resource group. The response includes certificate metadata such as the subject DN, expiry date, and serial number. The actual certificate and private key are never returned by the API; they are only accessible by mounting the secret in an execution or deployment.

Prerequisites

You've created at least one mTLS Certificate Secret. For more information, see Create an mTLS Certificate Secret [page 722].

Procedure

Submit a GET request to the endpoint `$AI_API_URL/v2/admin/mtlsCertificateSecrets`:

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Sample Code

```
curl --location --request GET "$AI_API_URL/v2/admin/mtlsCertificateSecrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'AI-Resource-Group: default'
```

Results

The response contains a list of mTLS certificate secret metadata objects.

Output Code

```
200 OK
{
    "count": 2,
    "resources": [
        {
            "name": "my-mtls-cert",
            "createdAt": "2026-03-01T12:00:00Z",
            "subjectDN": "...",
            "locality": "<locality-hash>",
            "commonName": "my-service-client",
            "expiresAt": "2024-10-01T12:00:00Z",
            "serialNumber": "aabbccdd..."
        },
        {
            "name": "another-cert",
            "createdAt": "2026-03-02T12:00:00Z",
            "subjectDN": "...",
            "locality": "<locality-hash>",
            "commonName": "another-cert",
            "expiresAt": "2024-10-02T12:00:00Z",
            "serialNumber": "eeff0011..."
        }
    ]
}
```

1.6.4.4 Get Details of an mTLS Certificate Secret

Retrieve metadata for a single mTLS certificate secret. The response includes certificate metadata such as the subject DN, expiry date, and serial number. The actual certificate and private key are never returned by the API; they are only accessible by mounting the secret in an execution or deployment.

Prerequisites

You've created at least one mTLS Certificate Secret. For more information, see Create an mTLS Certificate Secret [page 722].

Procedure

Submit a GET request to the endpoint `$AI_API_URL/v2/admin/mtlsCertificateSecrets`:

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Sample Code

```
curl --location --request GET "$AI_API_URL/v2/admin/mtlsCertificateSecrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'AI-Resource-Group: default'
```

Results

The response contains certificate metadata. No sensitive information is revealed.

Output Code

```
200 OK
{
    "name": "my-mtls-cert",
    "createdAt": "2024-07-01T12:00:00Z",
    "subjectDN": "C=DE, O=SAP SE, OU=SAP Cloud Platform Clients,
OU=<subaccount-ou>, L=<locality-hash>, CN=my-service-client",
    "locality": "<locality-hash>",
    "commonName": "my-service-client",
    "expiresAt": "2024-10-01T12:00:00Z",
    "serialNumber": "aabbccdd..."
}
```

1.6.4.5 Rotate an mTLS Certificate Secret

Rotate an mTLS certificate secret to obtain a new certificate with the same subject DN. The old certificate remains valid until its original expiry date.

Prerequisites

You've created at least one mTLS Certificate Secret. For more information, see Create an mTLS Certificate Secret [page 722].

Context

Certificate rotation generates a new certificate and private key for the secret while preserving the subject DN. Because the subject DN doesn't change, you don't need to update the trust configuration on your external target service.

After rotation, the secret is updated with the new certificate and key. Running workloads that have mounted the secret as a volume will see the new certificate files after a short propagation delay. If your workload reads the certificate at startup and keeps it in memory, you must restart the workload to pick up the rotated certificate.

> ⚠ Restriction
>
> mTLS certificate secrets are managed via the API and are available through SAP AI Core only.

> → Tip
>
> You can rotate a secret multiple times. Each rotation produces a new certificate with a unique serial number. The previous certificate isn't revoked and remains valid until its expiry date. This overlap allows you to rotate without downtime.

Procedure

Send a PATCH request to the endpoint `$AI_API_URL/v2/admin/mtlsCertificateSecrets`.

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

> {·} Sample Code

```
curl --location --request PATCH "$AI_API_URL/v2/admin/mtlsCertificateSecrets"
\
--header "Authorization: Bearer $TOKEN" \
--header 'AI-Resource-Group: default'
```

Results

Output Code

```
200 OK
{
    "name": "my-mtls-cert",
    "message": "...",
    "subjectDN": "C=DE, O=SAP SE, OU=SAP Cloud Platform Clients,
OU=<subaccount-ou>, L=<locality-hash>, CN=my-service-client"
}
```

The `subjectDN` remains unchanged. You can verify the rotation by calling the `GET` endpoint and checking that `serialNumber` has changed and `expiresAt` has been extended. For more information, see Get Details of an mTLS Certificate Secret [page 727].

1.6.4.6 Delete an mTLS certificate secret

Delete an mTLS certificate secret and retrieve the certificate's subject DN so that you can revoke trust on external services.

Prerequisites

To get the secret name, see List mTLS Certificate Secrets [page 726].

Context

You've completed the initial setup. For more information, see Initial Setup.

When you delete an mTLS certificate secret, the response includes the `subjectDN` of the certificate. Use this subject DN to revoke trust on the external service where you previously configured it.

If you do not revoke trust, any unexpired certificate issued for the same subject DN (for example, a certificate from before rotation) may still authenticate successfully.

Procedure

1. Send a DELETE request to the endpoint `$AI_API_URL/v2/admin/mtlsCertificateSecrets/$SECRET_NAME` .
2. Include the `AI-Resource-Group` header to specify the resource group.

3. Run the following request:

```
curl --location --request DELETE "$AI_API_URL/v2/admin/mtlsCertificateSecrets/
$SECRET_NAME" \
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: default"
```

Results

The service returns HTTP **200 OK** and the subject DN of the deleted certificate.

```
200 OK
{
    "name": "my-mtls-cert",
    "message": "...",
    "subjectDN": "C=DE, O=SAP SE, OU=SAP Cloud Platform Clients, OU=<subaccount-
ou>, L=<locality-hash>, CN=my-service-client"
}
```

> **Note**
>
> After deleting the secret, remove the trust configuration from your external service. The certificate may remain valid until its original expiry date even though the secret has been deleted from SAP AI Core.

Get a Secret Using a Third-Party API Platform

Procedure

1. Send a GET request to the endpoint `$AI_API_URL/v2/admin/mtlsCertificateSecrets/$SECRET_NAME`.
2. As the request body, select the *none* radio button.
3. Set the header `AI-Resource-Group` to the name of the resource group.
4. Send the request.

Results

> **Output Code**
>
> ```
> 200 OK
> {
>     "name": "my-mtls-cert",
>     "message": "...",
>     "subjectDN": "..."
> ```

```
}
```

1.6.5 Prompt Registry

Manage the life cycle of your prompts, from design to runtime.

Prompt registry integrates prompt templates into SAP AI Core, making them discoverable across your applications and orchestration.

It reduces the complexity of dealing with prompt templates and leveraging integration capabilities.

The You can use prompt registry to manage the lifecycle of the following:

- Prompt templates. For more information, see Prompt Template Management [page 732].
- Orchestration configs. For more information, see Orchestration Config Management [page 749].

Related Information

Access the Prompt Registry using the SAP AI SDK for Java
Access the Prompt Registry using the SAP AI SDK for JavaScript

1.6.5.1 Prompt Template Management

Manage the life cycle of your prompts, from design to runtime.

Prompt registry integrates prompt templates into SAP AI Core, making them discoverable across your applications and orchestration.

It reduces the complexity of dealing with prompt templates and leveraging integration capabilities.

The Prompt Registry consists of the following interfaces:

- The imperative API supports full CRUD operations and is recommended for refining prompt templates in design-time use cases. Iterations are tracked and can be viewed in the history endpoint. For more information, see Create a Prompt Template (Imperative) [page 733].
- The declarative API utilizes SAP AI Core applications and is recommended for runtime application use cases and CI/CD pipelines. You can manage your prompt templates in a git repository declaratively. Your commits are automatically synchronized with the Prompt Registry For more information, see Create a Prompt Template (Declarative) [page 735].

The interfaces can all be consumed either directly via the API or through Orchestration.

1.6.5.1.1 Create a Prompt Template (Imperative)

Context

> → Recommendation
>
> The imperative API supports full CRUD operations and is recommended for refining prompt templates in design-time use cases. Iterations are tracked and can be viewed in the history endpoint.

You can create a reusable prompt for a specific use case, including placeholders that are filled later.

By default, prompt templates are handled on the main-tenant level. To handle them on resource-group level, add the `AI-Resource-Group-Scope` and `AI-Resource-Group` headers to your requests. For example:

```
--header 'AI-Resource-Group-Scope: true'
--header 'AI-Resource-Group: <resource group>'
```

> ⓘ Note
>
> For CaaS users, the resource-group-id is extracted automatically, and doesn't need to be provided in the headers.

> ⓘ Note
>
> CaaS users only have write access to prompt templates on a resource-group level.

Procedure

Create a prompt template by sending a POST request to endpoint `$AI_API_URL/v2/lm/promptTemplates`.

> {·} Sample Code

```
curl -X POST "$AI_API_URL/v2/lm/promptTemplates"\
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $TOKEN" \
--data '{
    "name": "example-prompt-template",
    "version": "0.0.1",
    "scenario": "categorization",
    "spec": {
      "template": [
        {
          "role": "system",
          "content": "You classify input text into the two following
categories: {{?categories}}"
        },
        {
          "role": "user",
          "content": "{{?inputExample}}"
```

```
        }
      ],
      "defaults": {
        "categories": "Finance, Tech, Sports"
      },
      "additional_fields": {
        "modelParams": {
          "temperature": 0.7,
          "max_tokens": 100
        },
        "modelGroup": "chat"
      }
    }
  }'
```

- The `defaults` and `additional_fields` fields are optional.
- The `additional_fields` field is unstructured and can be used to store metadata or configuration objects.

{·} Output Code

```
{
  "id": "9y02-ha9x-92b1-4255",
  "name": "example-prompt-template",
  "versionId": "0.0.1",
  "message": "Prompt created successfully."
}
```

Next Steps

You can iterate over the prompt template and make changes to it.

You can save your changes as a new version. The latest iteration is always the head version and is marked with `isVersionHead: true`.

→ Tip

You'll need to update the version number when you want to create a new version, and the version entry must be compliant with semantic versioning (SemVer).

You can check the change history for the template. For more information, see Get Prompt Template History [page 739].

1.6.5.1.2 Create a Prompt Template (Declarative)

Prerequisites

- You've created and synced a git repository. For more information, see Add a Git Repository [page 706].
- You've created an application that points to the prompt templates in your git repository. For more information, see Create an Application.

Context

> → Recommendation
>
> The declarative API utilizes SAP AI Core applications and is recommended for runtime application use cases and CI/CD pipelines. You can manage your prompt templates in a git repository declaratively. Your commits are automatically synchronized with the prompt registry.

> ⓘ Note
>
> Declarative prompt templates are always handled on the main-tenant level.

Procedure

Create a prompt template and push it to your git repository. The file name must be in the format `<name>.prompttemplate.ai.sap.yaml` to be recognized.

The following formats are supported:

- yaml

> {·} Sample Code

```
name: simple
version: 0.0.1
scenario: my-scenario
spec:
  template:
    - role: "system"
      content: "{{ ?instruction }}"
    - role: "user"
      content: "Some more {{ ?user_input }}"
  defaults:
    instruction: "default instruction"
  additional_fields:
    isDev: true
    validations:
      required: true
```

```
    blockedModels:
      - name: "gpt-4"
        versions: "gpt-4-vision"
      - name: "gpt-4o"
        versions: "*"
```

- The `defaults` and `additional_fields` fields are optional.
- The `additional_fields` field is unstructured and can be used to store metadata or configuration objects.

Your template syncs automatically. After a few minutes, you'll be able to verify your template by sending a GET request to the endpoint `$AI_API_URL/v2/lm/promptTemplates`.

{·} Sample Code

```
{
    "count": 3,
    "resources": [
        {
            "id": "a460d210-df38-4867-9535-7a556701a4b0",
            "name": "simple",
            "version": "0.0.1",
            "scenario": "my-scenario",
            "creationTimestamp": "2024-08-18T14:50:17.157000",
            "managedBy": "declarative",
            "isVersionHead": true
        },
        {...}
    ]
}
```

The prompt template is marked as `managedBy`: `<declarative>`. Declaratively managed prompt templates are always the head version and can't be edited with the imperative API.

Next Steps

You use your prompt template at runtime. For more information, see Use a Prompt Template [page 741].

You can change your templates locally and push them to git.

→ Tip

For declaratively managed templates, use the history capabilities in git to track your changes.

1.6.5.1.3 Get a Prompt Template

Prerequisites

- You have created a prompt template. For more information, see Create a Prompt Template (Imperative) [page 733] and Create a Prompt Template (Declarative) [page 735].

Context

You can retrieve a prompt template by ID, or by the combination of name, scenario, and version.

Prompt templates can also be retrieved and consumed in orchestration. For more information, see Templating [page 344].

> **Note**
>
> Retrieval by ID offers immutability; guaranteeing that the prompt template behind the ID will not change.
>
> Retrieval by name, scenario, and version are not immutable, and the latest iteration of the prompt template is retrieved.

By default, prompt templates are handled on the main-tenant level. To handle them on resource-group level, add the `AI-Resource-Group-Scope` and `AI-Resource-Group` headers to your requests. For example:

```
--header 'AI-Resource-Group-Scope: true'
--header 'AI-Resource-Group: <resource group>'
```

> **Note**
>
> For CaaS users, the resource-group-id is extracted automatically, and doesn't need to be provided in the headers.

Procedure

Send a GET request to the endpoint `$AI_API_URL/v2/lm/promptTemplates`, and include the information for your chosen retrieval method.

> **Sample Code**

```
curl -X GET "$AI_API_URL/v2/lm/promptTemplates?name=example-prompt-
template&scenario=categorization&version=0.0.1" \
  -Header "Authorization: Bearer <your_auth_token>"
```

Sample Code

```
curl -X GET "$AI_API_URL/v2/lm/promptTemplates/{{promptTemplateId}}" \
-Header "Authorization: Bearer <your_auth_token>"
```

Pagination

The list endpoint supports pagination using the `$top` and `$skip` query parameters. This is useful when you have a large number of prompt templates and want to retrieve them in pages.

Note

If the result set is too large and no pagination parameters are supplied, the API returns a `413 Payload Too Large` error. Use `$top` and `$skip` to paginate through the results.

The following parameters are supported:

Parameter	Description	Default	Range
`$top`	Number of results to return per page	Full result set	1–500
`$skip`	Number of results to skip before returning the page	0	≥ 0

The response includes a `Link` header with navigation links when more pages are available:

- `rel=next` — URL for the next page of results.
- `rel=prev` — URL for the previous page of results (when applicable).

The `count` field in the response body always reflects the total number of matching prompt templates (not just the current page).

Sample Code

The following code retrieves the first page of 10 prompt templates:

```
curl -X GET "$AI_API_URL/v2/lm/promptTemplates?$top=10&$skip=0" \
  -Header "Authorization: Bearer <your_auth_token>"
```

The following code retrieves the second page of 10 prompt templates:

```
curl -X GET "$AI_API_URL/v2/lm/promptTemplates?$top=10&$skip=10" \
  -Header "Authorization: Bearer <your_auth_token>"
```

The following code combines filters with pagination:

```
curl -X GET "$AI_API_URL/v2/lm/promptTemplates?
scenario=categorization&$top=20&$skip=0" \
  -Header "Authorization: Bearer <your_auth_token>"
```

Example response with pagination (second page of 10, total 25 templates):

Output Code

```
HTTP/1.1 200 OK
Link: <$AI_API_URL/v2/lm/promptTemplates?$top=10&$skip=20>; rel="next",
<$AI_API_URL/v2/lm/promptTemplates?$top=10&$skip=0>; rel="prev"
{
  "count": 25,
  "resources": [
    {
      "id": "8y02-ha9x-92b1-4255",
      "name": "example-prompt-template",
      "version": "0.0.1",
      "scenario": "categorization",
      "managedBy": "imperative",
      "isVersionHead": true,
      "creationTimestamp": "2021-09-29T14:00:00Z"
    }
  ]
}
```

1.6.5.1.4 Get Prompt Template History

Context

You can list the history of edits to prompt templates, for imperatively managed prompt templates only.

By default, prompt templates are handled on the main-tenant level. To handle them on resource-group level, add the `AI-Resource-Group-Scope` and `AI-Resource-Group` headers to your requests. For example:

```
--header 'AI-Resource-Group-Scope: true'
--header 'AI-Resource-Group: <resource group>'
```

Note

For CaaS users, the resource-group-id is extracted automatically, and doesn't need to be provided in the headers.

Procedure

Send a GET request to endpoint `$AI_API_URL/v2/lm/scenarios/{{scenarioId}}/promptTemplates/{{promptTemplateName}}/versions/{{versionId}}/history` and include the name, scenario, and version of your prompt template.

Sample Code

```
curl -X GET "$AI_API_URL/v2/lm/scenarios/{{scenarioId}}/promptTemplates/
{{promptTemplateName}}/versions/{{versionId}}/history" \
  -Header "Authorization: Bearer <your_auth_token>"
```

Output Code

```
{
  "count": 10,
  "resources": [
    {
      "id": "8y02-ha9x-92b1-4255",
      "name": "example-prompt-template",
      "version": "0.0.1",
      "scenario": "categorization",
      "managedBy": "imperative",
      "isVersionHead": true,
      "creationTimestamp": "2021-09-29T14:00:00Z",
      "spec": {
        "template": [
          {
            "role": "system",
            "content": "You classify input text into the two following
categories: {{?categories}}"
          },
          {
            "role": "user",
            "content": "{{?inputExample}}"
          }
        ],
        "defaults": {
          "categories": "Finance, Tech, Sports"
        },
        "additional_fields": {
          "modelParams": {
            "temperature": 0.7,
            "max_tokens": 100
          },
          "modelGroup": "chat"
        }
      }
    }
  ]
}
```

Note

Responses for requests on a resource-group level, will include a `resourceGroupId` field, holding the hashed `resource-group-id`, instead of the `managedBy` field.

Pagination

The history endpoint supports pagination using the `$top` and `$skip` query parameters.

Note

If the result set is too large and no pagination parameters are supplied, the API returns a `413 Payload Too Large` error. Use `$top` and `$skip` to paginate through the results.

The following parameters are supported:

Parameter	Description	Default	Range
`$top`	Number of results to return per page	Full result set	1–500
`$skip`	Number of results to skip before returning the page	0	≥ 0

The response includes a `Link` header with navigation links when more pages are available:

- `rel=next` — URL for the next page of results.
- `rel=prev` — URL for the previous page of results (when applicable).

The `count` field in the response body always reflects the total number of matching prompt templates (not just the current page).

Sample Code

The following code retrieves the first page of 5 history entries:

```
curl -X GET "$AI_API_URL/v2/lm/scenarios/{{scenarioId}}/promptTemplates/
{{promptTemplateName}}/versions/{{versionId}}/history?$top=5&$skip=0" \
  -Header "Authorization: Bearer <your_auth_token>"
```

The following code retrieves the next page of 5 history entries:

```
curl -X GET "$AI_API_URL/v2/lm/scenarios/{{scenarioId}}/promptTemplates/
{{promptTemplateName}}/versions/{{versionId}}/history?$top=5&$skip=5" \
  -Header "Authorization: Bearer <your_auth_token>"
```

1.6.5.1.5 Use a Prompt Template

Prerequisites

- You have created a prompt template. For more information, see Create a Prompt Template (Imperative) [page 733].

Context

By default, prompt templates are handled on the main-tenant level. To handle them on resource-group level, add the `AI-Resource-Group-Scope` and `AI-Resource-Group` headers to your requests. For example:

```
--header 'AI-Resource-Group-Scope: true'
--header 'AI-Resource-Group: <resource group>'
```

> **Note**
>
> For CaaS users, the resource-group-id is extracted automatically, and doesn't need to be provided in the headers.

> **Note**
>
> CaaS users only have write access to prompt templates on a resource-group level.

Procedure

You can fill a prompt template by ID, or by the combination of name, scenario, and version.

- To fill a prompt template by ID, send a POST request to the endpoint `$AI_API_URL/v2/lm/promptTemplates/{{promptTemplateId}}/substitution` and add your variable values in the body.
- **Sample Code**

```
curl -X POST "$AI_API_URL/v2/lm/promptTemplates/{{promptTemplateId}}/
substitution" \
     -H "Authorization: Bearer <your_auth_token>" \
     -H "Content-Type: application/json" \
     -d '{
           "inputParams": {
             "inputExample": "Sample text"
           }
         }'
```

- To fill a prompt template by name, scenario, and version send a POST request to the endpoint `$AI_API_URL/v2/lm/scenarios/{{scenarioId}}/promptTemplates/{{promptTemplateName}}/versions/{{versionId}}/substitution` and add your variable values in the body.

  Sample Code

```
curl -X POST "$AI_API_URL/v2/lm/scenarios/{{scenarioId}}/promptTemplates/
{{promptTemplateName}}/versions/{{versionId}}/substitution" \
     -H "Authorization: Bearer <your_auth_token>" \
     -H "Content-Type: application/json" \
     -d '{
           "inputParams": {
             "inputExample": "Sample text"
           }
         }'
```

To return a response containing the full prompt template definition, include the optional query parameter `metadata = <true>`

1.6.5.1.6 Import a Prompt Template

Prerequisites

- You have created and exported a prompt template. For more information, see Create a Prompt Template (Imperative) [page 733] and Export a Prompt Template [page 744].

Context

You can import a declarative prompt template as a single file export in yaml format.

By default, prompt templates are handled on the main-tenant level. To handle them on resource-group level, add the `AI-Resource-Group-Scope` and `AI-Resource-Group` headers to your requests. For example:

```
--header 'AI-Resource-Group-Scope: true'
--header 'AI-Resource-Group: <resource group>'
```

> **Note**
>
> For CaaS users, the resource-group-id is extracted automatically, and doesn't need to be provided in the headers.

> **Note**
>
> CaaS users only have write access to prompt templates on a resource-group level.

Procedure

Send a POST request to the endpoint `$AI_API_URL/v2/lm/promptTemplates/import`, and include the name of your file in the `file` parameter.

> **Sample Code**

```
curl -X POST "$AI_API_URL/v2/lm/promptTemplates/import" \
     -Header "Authorization: Bearer <your_auth_token>" \
     -Header "Content-Type: multipart/form-data" \
     -File "file=@/path/to/your/file.yaml"
```

> **Output Code**

```
{
  "message": "Prompt <prompt-name> for version <version-number> imported
successfully.",
```

```
  "resource": {
    "name": "example-prompt-template",
    "version": "0.0.1",
    "scenario": "job-description",
    "spec": {
      "template": [
        {
          "role": "system",
          "content": "You classify input text into the two following
categories: {{?categories}}"
        },
        {
          "role": "user",
          "content": "{{?inputExample}}"
        }
      ],
      "defaults": {
        "categories": "Finance, Tech, Sports"
      },
      "additional_fields": {
        "modelParams": {
          "temperature": 0.7,
          "max_tokens": 100
        },
        "modelGroup": "chat"
      }
    }
  }
}
```

1.6.5.1.7 Export a Prompt Template

Prerequisites

- You have created a prompt template. For more information, see Create a Prompt Template (Imperative) [page 733].

Context

You can export a prompt template as a single file export in declarative compatible yaml format.

By default, prompt templates are handled on the main-tenant level. To handle them on resource-group level, add the `AI-Resource-Group-Scope` and `AI-Resource-Group` headers to your requests. For example:

```
--header 'AI-Resource-Group-Scope: true'
--header 'AI-Resource-Group: <resource group>'
```

> **Note**
>
> For CaaS users, the resource-group-id is extracted automatically, and doesn't need to be provided in the headers.

Procedure

Send a GET request to the endpoint `$AI_API_URL/v2/lm/promptTemplates/{{promptTemplateId}}/export`, and include the name of your file in the `output` parameter.

Sample Code

```
curl -X GET "$AI_API_URL/v2/lm/promptTemplates/{{promptTemplateId}}/export" \
-Header "Authorization: Bearer <your_auth_token>" \
--output "exported_prompt_template.yaml"
```

Output Code

```
name: example-prompt-template
version: 0.0.1
scenario: job-description
spec:
    template:
        - role: 'system'
        content: 'You classify input text into the two following categories:
{{?categories}}'
        - role: 'user'
        content: '{{?inputExample}}'
    defaults:
        categories: Finance, Tech, Sports
    additional_fields:
        modelParams:
            temperature: 0.7
            max_tokens: 100
        modelGroup: chat
        collection: abc
```

1.6.5.1.8 Delete a Prompt Template

Prerequisites

- You have created a prompt template. For more information, see Create a Prompt Template (Imperative) [page 733].
- You know the template ID of the prompt template that you want to delete. For more information, see Get a Prompt Template [page 737].

Context

Delete a specific version of the prompt template, for imperatively managed prompt templates only.

By default, prompt templates are handled on the main-tenant level. To handle them on resource-group level, add the `AI-Resource-Group-Scope` and `AI-Resource-Group` headers to your requests. For example:

```
--header 'AI-Resource-Group-Scope: true'
--header 'AI-Resource-Group: <resource group>'
```

> **Note**
>
> For CaaS users, the resource-group-id is extracted automatically, and doesn't need to be provided in the headers.

> **Note**
>
> CaaS users only have write access to prompt templates on a resource-group level.

Procedure

Send a DELETE request to the endpoint `$AI_API_URL/v2/lm/promptTemplates/{{promptTemplateId}}` and include the template ID of the prompt you want to delete.

> **Output Code**

```
{
  "message": "Prompt deleted successfully."
}
```

1.6.5.1.9 Delete Prompt Template History

Prerequisites

- You've created a prompt template. For more information, see Create a Prompt Template (Imperative) [page 733].
- You know the name, scenario, and version of the prompt template whose history you want to delete. For more information, see Get Prompt Template History [page 739].

Context

Delete all non-head versions (history) of a specific prompt template. The current head version is preserved. For imperatively managed prompt templates only.

By default, prompt templates are handled on the main-tenant level. To handle them on resource-group level, add the `AI-Resource-Group-Scope` and `AI-Resource-Group` headers to your requests. For example:

```
--header 'AI-Resource-Group-Scope: true'
--header 'AI-Resource-Group: <resource group>'
```

> **Note**
>
> For CaaS users, the resource-group-id is extracted automatically, and doesn't need to be provided in the headers.

> **Note**
>
> CaaS users only have write access to prompt templates on a resource-group level.

Procedure

Send a DELETE request to the endpoint `$AI_API_URL/v2/lm/scenarios/{{scenarioId}}/promptTemplates/{{promptTemplateName}}/versions/{{versionId}}/history` and include the scenario, name, and version of the prompt template whose history you want to delete.

```
curl -X DELETE "$AI_API_URL/v2/lm/scenarios/{{scenarioId}}/promptTemplates/
{{promptTemplateName}}/versions/{{versionId}}/history" \
  --header "Authorization: Bearer <your_auth_token>"
```

Results

> **Output Code**

```
{
  "message": "Prompt template history deleted successfully.",
  "deleted": 3
}
```

1.6.5.1.10 Delete All Prompt Templates under a specific Scenario

Prerequisites

- You have created a prompt template. For more information, see Create a Prompt Template (Imperative) [page 733].
- You know the scenario ID of the scenario whose prompt templates you want to delete.

Context

Delete all prompt templates under a given scenario. This operation affects imperatively managed prompt templates only.

By default, prompt templates are handled on the main-tenant level. To handle them on resource-group level, add the `AI-Resource-Group-Scope` and `AI-Resource-Group` headers to your requests. For example:

```
--header 'AI-Resource-Group-Scope: true'
--header 'AI-Resource-Group: <resource group>'
```

> **Note**
>
> For CaaS users, the resource-group-id is extracted automatically, and doesn't need to be provided in the headers.

> **Note**
>
> CaaS users only have write access to prompt templates on a resource-group level.

Procedure

Send a DELETE request to the endpoint `$AI_API_URL/v2/lm/scenarios/{{scenarioId}}/promptTemplates` and include the scenario ID.

```
curl -X DELETE "$AI_API_URL/v2/lm/scenarios/{{scenarioId}}/promptTemplates" \
  --header "Authorization: Bearer <your_auth_token>"
```

Results

Output Code

```
{
  "message": "Prompt templates deleted successfully.",
  "deleted": 5
}
```

1.6.5.2 Orchestration Config Management

Manage the lifecycle of your orchestration configurations, from design to runtime.

You can store, version, and manage orchestration configs as reusable assets in the prompt registry. This makes them discoverable across your applications and orchestration workflows, enabling better governance and reusability.

Orchestration configs can reference existing prompt templates by ID or by name, scenario, and version, allowing you to build reusable orchestration workflows that leverage your prompt template library.

Stored orchestration configs can be referenced in LLM Orchestration completion requests, providing an alternative to inline configuration definitions.

The Prompt Registry consists of the following interfaces for orchestration configs:

- The imperative API supports full CRUD operations and is recommended for refining orchestration configs in design-time use cases. Iterations are tracked and can be viewed in the history endpoint. For more information, see Create an Orchestration Config (Imperative) [page 750].
- The declarative API utilizes SAP AI Core applications and is recommended for runtime application use cases and CI/CD pipelines. You can manage your orchestration configs in a git repository declaratively. Your commits are automatically synchronized with the Prompt Registry For more information, see Create an Orchestration Config (Imperative) [page 750].

The interfaces can all be consumed either directly via the API or through Orchestration.

Stored orchestration configs reduce the complexity of managing orchestration configurations by providing lifecycle management capabilities including versioning, history tracking, and import and export functionality.

The orchestration config endpoints are imperative API operations that support full CRUD functionality for both design-time and runtime use cases. Iterations are tracked and can be viewed in the history endpoint.

Note

Orchestration configs are only available through the imperative API and are managed at the tenant level.

1.6.5.2.1 Create an Orchestration Config (Imperative)

Context

> → Recommendation
>
> The imperative API supports full CRUD operations and is recommended for refining orchestration configs in design-time use cases. Iterations are tracked and can be viewed in the history endpoint.

You can create a reusable orchestration configuration for a specific use case, including module configurations for filtering, masking, grounding, and prompt templating.

Orchestration configs are handled at the tenant level and do not require resource group headers.

Orchestration configs can reference existing prompt templates in two ways:

- By template ID for immutable references
- By name, scenario, and version for flexible references that can point to the latest version

Procedure

Create an orchestration config by sending a POST request to endpoint `$AI_API_URL/v2/registry/v2/orchestrationConfigs`.

Set the following in the headers:

- `AI_API_URL`: the base URL of your SAP AI Core environment.
- `{{access_token}}`: Your access token for SAP AI Core

The following examples show only the `prompt_templating` module for simplicity. Orchestration configs can include all supported modules such as `filtering`, `masking`, `grounding`, and `translation` in addition to `prompt_templating`.

> ⓘ Note
>
> Use template references by ID when you need immutable references that won't change. Use references by name, scenario, and version when you want flexibility to update the underlying template while keeping the same orchestration config.

Example With Inline Prompt Template

> {·} Sample Code

```
curl -X POST "$AI_API_URL/v2/registry/v2/orchestrationConfigs" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $TOKEN" \
--data '{
    "name": "example-orchestration-config",
```

```
    "version": "0.0.1",
    "scenario": "customer-support",
    "spec": {
      "modules": {
        "prompt_templating": {
          "prompt": {
            "template": [
              {
                "role": "system",
                "content": "You are a helpful customer support assistant."
              },
              {
                "role": "user",
                "content": "{{?userQuery}}"
              }
            ],
            "defaults": {
              "userQuery": "How can I help you today?"
            }
          },
          "model": {
            "name": "<model>",
            "params": {
              "temperature": 0.7,
              "max_tokens": 500
             }
          }
        },
        "grounding": {
          "type": "document_grounding_service",
          "config": {
            "filters": [
              {
                "id": "filter1",
                "data_repositories": ["*"],
                "search_config": {},
                "data_repository_type": "vector"
              }
            ],
            "placeholders": {
              "input":  ["groundingInput"],
              "output": "groundingOutput"
            }
          }
        }
      }
    }
  }'
```

Example With Prompt Template Reference by ID

Sample Code

```
curl -X POST "$AI_API_URL/v2/registry/v2/orchestrationConfigs" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $TOKEN" \
--data '{
    "name": "example-orchestration-config",
    "version": "0.0.1",
    "scenario": "customer-support",
    "spec": {
      "modules": {
        "prompt_templating": {
          "prompt": {
            "template_ref": {
              "id": "<templateId>"
```

```
            }
          },
          "model": {
            "name": "model",
            "params": {
              "temperature": 0.7,
             }
          }
        },
        "grounding": {
          "type": "document_grounding_service",
          "config": {
            "filters": [
              {
                "id": "filter1",
                "data_repositories": ["*"],
                "search_config": {},
                "data_repository_type": "vector"
              }
            ],
            "placeholders": {
              "input":  ["groundingInput"],
              "output": "groundingOutput"
            }
          }
        }
      }
    }
  }'
```

Example With Scenario, Template Name, and Version Combination

Sample Code

```
curl -X POST "$AI_API_URL/v2/registry/v2/orchestrationConfigs" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $TOKEN" \
--data '{
    "name": "example-orchestration-config",
    "version": "0.0.1",
    "scenario": "customer-support",
    "spec": {
      "modules": {
        "prompt_templating": {
          "prompt": {
            "template_ref": {
              "scenario": "categorization",
              "name": "example-prompt-template",
              "version": "0.0.1"
            }
          },
          "model": {
            "name": "<model>",
            "params": {
              "temperature": 0.7,
              "max_tokens": 500
             }
          }
        },
        "grounding": {
          "type": "document_grounding_service",
          "config": {
            "filters": [
              {
                "id": "filter1",
                "data_repositories": ["*"],
```

```
                "search_config": {},
                "data_repository_type": "vector"
              }
            ],
            "placeholders": {
              "input":  ["groundingInput"],
              "output": "groundingOutput"
            }
          }
        }
      }
    }
  }'
```

Example of a Partial Orchestration Config

Orchestration configs that lack mandatory fields (such as model or prompt details) can still be created, but the mandatory fields must be provided at time of consumption in Orchestration. In the following example, the prompt is missing:

```
curl -X POST "$AI_API_URL/v2/registry/v2/orchestrationConfigs" \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $TOKEN" \
--data '{
    "name": "partial-orchestration-config",
    "version": "0.0.1",
    "scenario": "customer-support",
    "spec": {
      "modules": {
        "prompt_templating": {
          "model": {
            "name": "gpt-4o"
          }
        }
      }
    }
  }'
```

The following consumption example provides the mandatory prompt and an optional tool definition:

```
curl --request POST \
  --url $ORCHESTRATION_DEPLOYMENT_URL/v2/completion \
  --header "AI-Resource-Group: $RESOURCE_GROUP" \
  --header 'authorization: Bearer {{TOKEN}}' \
  --header 'content-type: application/json' \
  --data '{
  "config_ref": {
    "id": "b5ea5b32-0151-4608-98f6-d260dba6700d"
  },
  "config": {
    "modules": {
      "prompt_templating": {
        "prompt": {
          "template": [
            {
              "role": "user",
              "content": "What is the weather in Potsdam and in Toulouse in
Celsius?"
            }
          ],
```

```
          "defaults": {},
          "tools": [
            {
              "type": "function",
              "function": {
                "description": "Get weather for location",
                "name": "getCurrentWeather",
                "parameters": {
                  "$schema": "https://json-schema.org/draft/2020-12/schema",
                  "additionalProperties": false,
                  "type": "object",
                  "properties": {
                    "arg0": {
                      "type": "object",
                      "properties": {
                        "location": {
                          "type": "string"
                        },
                        "unit": {
                          "type": "string",
                          "enum": [
                            "C",
                            "F"
                          ]
                        }
                      },
                      "required": [
                        "location",
                        "unit"
                      ]
                    }
                  },
                  "required": [
                    "arg0"
                  ]
                },
                "strict": false
              }
            }
          ]
        }
      }
    }
  }
}'
```

Results

You will receive an output JSON containing details about your configuration, including an ID and success message.

Output Code

```
{
  "id": "<orchestrationConfigId>",
  "name": "example-orchestration-config",
  "version": "0.0.1",
  "scenario": "customer-support",
  "message": "Orchestration config created successfully."
}
```

Next Steps

You can iterate over the orchestration config and make changes to it.

You can save your changes as a new version. To create a new version, update the version number. Your entry must be compliant with semantic versioning (SemVer). The latest iteration is always the head version and is marked with `is_version_head: true`.

You can check the change history for the config. For more information, see Get Orchestration Configs [page 757].

1.6.5.2.2 Create an Orchestration Config (Declarative)

Prerequisites

- You've created and synced a git repository. For more information, see Add a Git Repository [page 706].
- You've created an application that points to the orchestration configs in your git repository. For more information, see Create an Application.

Context

> → Recommendation
>
> The declarative API utilizes SAP AI Core applications and is recommended for runtime application use cases and CI/CD pipelines. You can manage your orchestration configs in a git repository declaratively. Your commits are automatically synchronized with the prompt registry.

> ⓘ Note
>
> Declarative orchestration configs are always handled on the main-tenant level.

Procedure

Create a orchestration config and push it to your git repository. The file name must be in the format `<name>.orchestrationconfig.ai.sap.yaml` to be recognized.

The following formats are supported:

- yaml

Sample Code

```
name: simple
version: 0.0.1
scenario: my-scenario
spec:
  modules:
    prompt_templating:
      prompt:
        template:
        - role: system
          content: You are a helpful customer support assistant.
        - role: user
          content: "{{?userQuery}}"
        defaults:
          userQuery: How can I help you today?
      model:
        name: "<model>"
        params:
          temperature: 0.7
          max_tokens: 500
```

The `modules` field is required for orchestration configs. You can define additional configurations as needed for your orchestration workflow. For more information, see Orchestration Workflow V2 [page 382].

Your orchestration config syncs automatically. After a few minutes, you'll be able to verify your config by sending a GET request to the endpoint`$AI_API_URL/v2/registry/v2/orchestrationConfigs`.

Sample Code

```
{
  "count": 1,
  "resources": [
    {
      "id": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
      "name": "test-i342302",
      "version": "0.0.1",
      "scenario": "test-i342302-scenario",
      "creationTimestamp": "2026-02-02T10:00:00.000000",
      "managedBy": "declarative",
      "isVersionHead": true
    }
  ]
}
```

The orchestration config is marked as `managedBy`: `<declarative>`. Declaratively managed orchestration configs are always the head version and can't be edited with the imperative API.

Next Steps

You use your orchestration config at runtime. For more information, see Use an Orchestration Config in LLM Orchestration [page 762].

You can change your templates locally and push them to git.

→ Tip

For declaratively managed templates, use the history capabilities in git to track your changes.

1.6.5.2.3 Get Orchestration Configs

Prerequisites

You've created an orchestration config. For more information, see Create an Orchestration Config (Imperative) [page 750] and Create an Orchestration Config (Declarative) [page 755].

Context

You can list all orchestration configs or retrieve a specific orchestration config by ID, or by the combination of name, scenario, and version.

You can also retrieve and consume orchestration configs in orchestration completion requests. For more information, see Use an Orchestration Config in LLM Orchestration [page 762].

> **Note**
>
> When you retrieve a configuration by ID, the result is immutable. The configuration associated with that ID doesn't change.
>
> When you retrieve a configuration by name, scenario, and version, the result isn't immutable. The system returns the latest available iteration of the matching orchestration configuration, which can change over time.
>
> Orchestration configurations are handled at tenant level so you don't need to specify resource group headers when accessing them.

Procedure

Send a GET request to the endpoint `{{apiurl}}/v2/registry/v2/orchestrationConfigs`, and include the information for your chosen retrieval method.

Set the following in the headers:

- `AI_API_URL`: the base URL of your SAP AI Core environment.
- `{{access_token}}`: Your access token for SAP AI Core

List All Orchestration Configs

> **Sample Code**

```
curl -X GET "{{apiurl}}/v2/registry/v2/orchestrationConfigs" \
  -H "Authorization: Bearer <your_auth_token>"
```

Get by Name, Scenario, and Version

Sample Code

```
curl -X GET "{{apiurl}}/v2/registry/v2/orchestrationConfigs?name=example-
orchestration-config&scenario=customer-support&version=0.0.1" \
  -H "Authorization: Bearer <your_auth_token>"
```

Get by ID

Sample Code

```
curl -X GET "{{apiurl}}/v2/registry/v2/orchestrationConfigs/
{{orchestrationConfigId}}" \
-H "Authorization: Bearer <your_auth_token>"
```

→ Tip

Use the `resolve_template_ref=true` query parameter to resolve template references and include the full template definition in the response.

Results

Output Code

```
{
  "count": 3,
  "resources": [
    {
      "id": "<orchestrationConfigId1>",
      "name": "example-orchestration-config",
      "version": "0.0.1",
      "scenario": "customer-support",
      "creation_timestamp": "2024-08-18T14:50:17.157000",
      "managed_by": "imperative",
      "is_version_head": true
    },
    {
      "id": "<orchestrationConfigId2>",
      "name": "another-orchestration-config",
      "version": "1.0.0",
      "scenario": "data-analysis",
      "creation_timestamp": "2024-08-19T10:30:45.123000",
      "managed_by": "declarative",
      "is_version_head": true
    },
    {
      "id": "<orchestrationConfigId2>",
      "name": "another-orchestration-config",
      "version": "1.0.0",
      "scenario": "data-analysis",
      "creation_timestamp": "2024-08-19T10:30:45.123000",
      "managed_by": "imperative",
      "is_version_head": true
    }
  ]
```

```
}
```

Pagination

The list endpoint supports pagination using the `$top` and `$skip` query parameters. These parameters are useful when you have a large number of prompt templates and want to retrieve them in pages.

> **Note**
>
> If the result set is too large and no pagination parameters are supplied, the API returns a `413 Payload Too Large` error. Use `$top` and `$skip` to paginate through the results.

The following parameters are supported:

Parameter	Description	Default	Range
`$top`	Number of results to return per page	Full result set	1–500
`$skip`	Number of results to skip before returning the page	0	≥ 0

The response includes a `Link` header with navigation links when more pages are available:

- `rel=next` — URL for the next page of results.
- `rel=prev` — URL for the previous page of results (when applicable).

The `count` field in the response body always reflects the total number of matching prompt templates (not just the current page).

> **Sample Code**
>
> The following code retrieves the first page of 10 orchestration configs:
>
> ```
> curl -X GET "$AI_API_URL/v2/registry/v2/orchestrationConfigs?$top=10&$skip=0"
> \
>   -Header "Authorization: Bearer <your_auth_token>"
> ```
>
> The following code retrieves the second page of 10 orchestration configs:
>
> ```
> curl -X GET "$AI_API_URL/v2/registry/v2/orchestrationConfigs?
> $top=10&$skip=10" \
>   -Header "Authorization: Bearer <your_auth_token>"
> ```
>
> The following code combines filters with pagination:
>
> ```
> curl -X GET "$AI_API_URL/v2/registry/v2/orchestrationConfigs?
> scenario=scenario=customer-support&$top=20&$skip=0" \
>   -Header "Authorization: Bearer <your_auth_token>"
> ```

Example response with pagination (second page of 10, total 25 configs):

Output Code

```
HTTP/1.1 200 OK
Link: <$AI_API_URL/v2/registry/v2/orchestrationConfigs?$top=10&$skip=20>;
rel="next", <$AI_API_URL/v2/registry/v2/orchestrationConfigs?
$top=10&$skip=0>; rel="prev"
{
  "count": 25,
  "resources": [
    {
      "id": "<orchestrationConfigId>",
      "name": "example-orchestration-config",
      "version": "0.0.1",
      "scenario": "customer-support",
      "creation_timestamp": "2024-08-18T14:50:17.157000",
      "managed_by": "imperative",
      "is_version_head": true
    }
  ]
}
```

1.6.5.2.4 Get Orchestration Config History

Context

You can list the history of edits to orchestration configs for imperatively managed configurations.

Orchestration configs are handled at the tenant level and do not require resource group headers.

Procedure

Send a GET request to endpoint `{{apiurl}}/v2/registry/v2/scenarios/{{scenarioId}}/orchestrationConfigs/{{orchestrationConfigName}}/versions/{{versionId}}/history` and include the name, scenario, and version of your orchestration config.

Set the following in the headers:

- `AI_API_URL`: the base URL of your SAP AI Core environment.
- `{{access_token}}`: Your access token for SAP AI Core

Sample Code

```
curl -X
GET "{{apiurl}}/v2/registry/v2/scenarios/{{scenarioId}}/orchestrationConfigs/
{{orchestrationConfigName}}/versions/{{versionId}}/history" \
  -H "Authorization: Bearer <your_auth_token>"
```

Output Code

```
{
  "count": 5,
  "resources": [
    {
      "id": "<orchestrationConfigId>",
      "name": "example-orchestration-config",
      "version": "0.0.1",
      "scenario": "customer-support",
      "managed_by": "imperative",
      "is_version_head": true,
      "creation_timestamp": "2021-09-29T14:00:00Z",
      "spec": {
        "modules": {
          "prompt_templating": {
            "prompt": {
              "template_ref": {
                "id": "<promptTemplateId>"
              }
            },
            "model": {
              "name": "<model>",
              "params": {
                "temperature": 0.7,
                "max_tokens": 500
              }
            }
          }
        }
      }
    }
  ]
}
```

Pagination

The history endpoint supports pagination using the `$top` and `$skip` query parameters.

Note

If the result set is too large and no pagination parameters are supplied, the API returns a `413 Payload Too Large` error. Use `$top` and `$skip` to paginate through the results.

The following parameters are supported:

Parameter	Description	Default	Range
`$top`	Number of results to return per page	Full result set	1–500
`$skip`	Number of results to skip before returning the page	0	≥ 0

The response includes a `Link` header with navigation links when more pages are available:

- `rel=next` — URL for the next page of results.

- `rel=prev` — URL for the previous page of results (when applicable).

The `count` field in the response body always reflects the total number of matching prompt templates (not just the current page).

> Sample Code
>
> The following code retrieves the first page of 5 history entries:
>
> ```
> curl -X GET "$AI_API_URL/v2/registry/v2/scenarios/$scenario/
> orchestrationConfigs/$name/versions/$version/history?$top=5&$skip=0" \
>   -Header "Authorization: Bearer <your_auth_token>"
> ```
>
> The following code retrieves the next page of 5 history entries:
>
> ```
> curl -X GET "$AI_API_URL/v2/registry/v2/scenarios/$scenario/
> orchestrationConfigs/$name/versions/$version/history?$top=5&$skip=5" \
>   -Header "Authorization: Bearer <your_auth_token>"
> ```

1.6.5.2.5 Use an Orchestration Config in LLM Orchestration

Prerequisites

You have created an orchestration config. For more information, see Create an Orchestration Config (Imperative) [page 750] and Create an Orchestration Config (Declarative) [page 755].

Context

Orchestration configs can be used in LLM Orchestration completion requests in the following ways:

1. **Inline configuration**: Define the configuration directly in the request
2. **Reference by ID**: Reference a stored config by its unique ID
3. **Reference by name, scenario, and version**: Reference a stored config by its metadata

Procedure

You can reference orchestration configs using the following methods:

Set the following in the headers:

- `ORCHESTRATION_DEPLOYMENT_URL`: your orchestration deployment url. For more information, see Get An Orchestration Deployment URL [page 7].

- `{{access_token}}`: Your access token for SAP AI Core

Method 1: Reference Orchestration Config by ID

Send a POST request to the LLM Orchestration completion endpoint with a config reference by ID.

```
curl -X POST "$ORCHESTRATION_DEPLOYMENT_URL/v2/completion" \
     -H "Authorization: Bearer <your_auth_token>" \
     -H "Content-Type: application/json" \
     -d '{
  "config_ref": {
    "id": "<orchestrationConfigId>"
  },
  "placeholder_values": {
    "userQuery": "What are the benefits of SAP AI Core?",
    "groundingInput": "How many minimum annual leave days do SAP full-time
employees have?"
  }
}'
```

> **Note**
>
> Placeholder keys must match the stored orchestration config exactly, and are case-sensitive.

Method 2: Reference Orchestration Config by Name, Scenario, and Version

Send a POST request to the LLM Orchestration completion endpoint with a config reference by metadata.

```
curl -X POST "$ORCHESTRATION_DEPLOYMENT_URL/v2/completion" \
     -H "Authorization: Bearer <your_auth_token>" \
     -H "Content-Type: application/json" \
     -d '{
  "config_ref": {
    "scenario": "customer-support",
    "name": "example-orchestration-config",
    "version": "0.0.1"
  },
  "placeholder_values": {
    "userQuery": "What are the benefits of SAP AI Core?",
    "groundingInput": "How many minimum annual leave days do SAP full-time
employees have?"
  },
  "messages_history": [
    {
      "role": "system",
      "content": "You are helping with technical integration questions."
    }
  ]
}'
```

> **Note**
>
> Placeholder keys must match the stored orchestration config exactly, and are case-sensitive.

Method 3: Inline Configuration (for comparison)

You can also use orchestration configurations inline without storing them first.

```
curl -X POST "$ORCHESTRATION_DEPLOYMENT_URL/v2/completion" \
     -H "Authorization: Bearer <your_auth_token>" \
     -H "Content-Type: application/json" \
     -d '{
```

```
          "config": {
            "modules": {
              "prompt_templating": {
                "prompt": {
                  "template": [
                    {
                      "role": "user",
                      "content": "{{?userQuery}}"
                    }
                  ]
                },
                "model": {
                  "name": "<model>"
                }
              }
            }
          },
          "placeholder_values": {
            "userQuery": "What is SAP AI Core?"
          }
        }'
```

> **Note**
>
> Use stored orchestration configs (methods 1 and 2) when you have reusable configurations that multiple applications or workflows need to access. Use inline configurations (method 3) for one-off or experimental requests.

> **Note**
>
> When referencing orchestration configs in LLM Orchestration:
>
> - Reference by ID provides immutability meaning that the config behind the ID will never change
> - Reference by name, scenario, and version provides flexibility meaning that you get the latest iteration of that version

1.6.5.2.6 Export an Orchestration Config

Prerequisites

You have created an orchestration config. For more information, see Create an Orchestration Config (Imperative) [page 750] and Create an Orchestration Config (Declarative) [page 755].

Context

You can export an orchestration config as a single file export in YAML format.

Orchestration configs are handled at the tenant level and do not require resource group headers.

Procedure

Send a GET request to the endpoint `{{apiurl}}/v2/registry/v2/orchestrationConfigs/{{orchestrationConfigId}}/export`, and include the name of your file in the output parameter.

Set the following in the headers:

- `AI_API_URL`: the base URL of your SAP AI Core environment.
- `{{access_token}}`: Your access token for SAP AI Core

Sample Code

```
curl -X GET "{{apiurl}}/v2/registry/v2/orchestrationConfigs/
{{orchestrationConfigId}}/export" \
-H "Authorization: Bearer <your_auth_token>" \
--output "exported_orchestration_config.yaml"
```

Output Code

```
name: example-orchestration-config
version: 0.0.1
scenario: customer-support
spec:
  modules:
    prompt_templating:
      prompt:
        template_ref:
          id: <promptTemplateId>
      model:
        name: <model>
        params:
          temperature: 0.7
          max_tokens: 500
```

1.6.5.2.7 Import an Orchestration Config

Prerequisites

You have created an orchestration config. For more information, see Create an Orchestration Config (Imperative) [page 750], Create an Orchestration Config (Declarative) [page 755] and Export an Orchestration Config [page 764].

Prerequisites

You have created and exported an orchestration config. For more information, see Create an Orchestration Config (Imperative) [page 750]

Context

You can import an orchestration config as a single file export in YAML format.

Orchestration configs are handled at the tenant level and do not require resource group headers.

Procedure

Send a POST request to the endpoint `{{apiurl}}/v2/registry/v2/orchestrationConfigs/import`, and include the name of your file in the file parameter.

Set the following in the headers:

- `AI_API_URL`: the base URL of your SAP AI Core environment.
- `{{access_token}}`: Your access token for SAP AI Core

Sample Code

```
curl -X POST "{{apiurl}}/v2/registry/v2/orchestrationConfigs/import" \
     -H "Authorization: Bearer <your_auth_token>" \
     -H "Content-Type: multipart/form-data" \
     -F "file=@/path/to/your/orchestration-config.yaml"
```

Output Code

```
{
  "message": "Orchestration config example-orchestration-config for version
0.0.1 imported successfully.",
  "id": "<orchestrationConfigId>",
  "name": "example-orchestration-config",
  "version": "0.0.1",
  "scenario": "customer-support"
}
```

1.6.5.2.8 Delete an Orchestration Config

Prerequisites

You have created an orchestration config. For more information, see Create an Orchestration Config (Imperative) [page 750].

You know the config ID of the orchestration config that you want to delete. For more information, see Get Orchestration Configs [page 757].

Context

Delete a specific version of the orchestration config for imperatively managed configurations.

Orchestration configs are handled at the tenant level and do not require resource group headers.

Procedure

Send a DELETE request to the endpoint `{{apiurl}}/v2/registry/v2/orchestrationConfigs/{{orchestrationConfigId}}` and include the config ID of the orchestration config you want to delete.

Set the following in the headers:

- `AI_API_URL`: the base URL of your SAP AI Core environment.
- `{{access_token}}`: Your access token for SAP AI Core

Sample Code

```
curl -X DELETE "{{apiurl}}/v2/registry/v2/orchestrationConfigs/
{{orchestrationConfigId}}" \
-H "Authorization: Bearer <your_auth_token>"
```

Output Code

```
{
  "message": "Orchestration config deleted successfully."
}
```

1.6.5.2.9 Delete Orchestration Config History

Prerequisites

- You have created an orchestration config. For more information, see Create an Orchestration Config (Imperative) [page 750].
- You know the name, scenario, and version of the orchestration config whose history you want to delete. For more information, see Get Orchestration Config History [page 760].

Context

Delete all non-head versions (history) of a specific orchestration config identified by its scenario, name, and version. The current head version is preserved. For imperatively managed orchestration configs only.

Orchestration configs are handled at the tenant level and do not require resource group headers.

Procedure

Send a DELETE request to the endpoint `$AI_API_URL/v2/registry/v2/scenarios/{{scenarioId}}/orchestrationConfigs/{{orchestrationConfigName}}/versions/{{versionId}}/history` and include the scenario, name, and version of the orchestration config whose history you want to delete.

```
curl -X DELETE "$AI_API_URL/v2/registry/v2/scenarios/{{scenarioId}}/
orchestrationConfigs/{{orchestrationConfigName}}/versions/{{versionId}}/history"
\
  --header "Authorization: Bearer <your_auth_token>"
```

Results

Output Code

```
{
  "message": "Orchestration config history deleted successfully.",
  "deleted": 3
}
```

1.6.5.2.10 Delete All Orchestration Configs under a specific Scenario

Prerequisites

- You have created an orchestration config. For more information, see Create an Orchestration Config (Imperative) [page 750].
- You know the scenario ID of the scenario whose orchestration configs you want to delete.

Context

Delete all orchestration configs under a given scenario. This operation affects imperatively managed orchestration configs only.

Orchestration configs are handled at the tenant level and do not require resource group headers.

Procedure

Send a DELETE request to the endpoint `$AI_API_URL/v2/registry/v2/scenarios/{{scenarioId}}/orchestrationConfigs` and include the scenario ID.

```
curl -X DELETE "$AI_API_URL/v2/registry/v2/scenarios/{{scenarioId}}/
orchestrationConfigs" \
  --header "Authorization: Bearer <your_auth_token>"
```

Results

Output Code

```
{
  "message": "Orchestration configs deleted successfully.",
  "deleted": 5
}
```

1.6.6 Rate Limit Management

Quota-management APIs let you check and modify rate limits for generative AI model inference calls.

SAP AI Core implements rate-limiting controls that restrict the number of Requests Per Minute (RPM) for each model within a tenant environment. When these limits are exceeded, the system responds with a 429 error code.

The rate limits are model-specific and reset on a per-minute basis.

By default, all resource groups within a tenant operate under the same rate-limit threshold, although you can customize this behavior through specific configuration. Additionally, all versions of a particular model share the same configured rate-limit allocation.

For a tenant containing resource group (RG) 1 and resource group 2: if RG1 has a rate limit of 12k, and RG2 has a rate limit of 6k, the rate limit for the tenant must be at least 12k.

429 Too Many Requests

If an API returns HTTP status 429, the request is rejected because the system cannot process it due to load or quota constraints. This response can indicate rate limits or temporary backend capacity issues.

Common Causes and Mitigation

Cause	Scope	Mitigation
Tenant rate limit exceeded	All resource groups in tenant	Retry with backoff, distribute load, request quota increase
Resource group rate limit exceeded	Specific resource group	Retry with backoff, isolate workloads, request quota increase
Hyperscaler capacity exceeded	Infrastructure level	Retry later using `Retry-After`

> **Note**
>
> A 429 response does not always mean a rate limit is exceeded. It can also indicate temporary back-end saturation.

Causes

Tenant-Level Rate Limits

SAP AI Core enforces per-tenant rate limits measured in RPM.

All resource groups share this limit by default. When the combined request rate exceeds the threshold, subsequent requests return 429 until the next minute window.

- Limits are model-specific
- All model versions share the same limit

- Limits reset every minute

Resource Group-Level Rate Limits

You can configure rate limits for individual resource groups.

If configured, requests from that resource group use its dedicated limit instead of the shared tenant quota.

Hyperscaler and Service Provider Limits

SAP AI Core forwards requests to underlying infrastructure such as Azure OpenAI and SAP services.

These services may reject requests when their capacity limits are reached, independent of SAP AI Core quotas.

Response Headers

Rate Limit Response Headers

Header	Description
`Retry-After`	Time in seconds to wait before retrying

Mitigation Strategies

> **Note**
>
> A `429` response indicates a temporary condition, not a system failure.

Implement Retry Logic with Exponential Backoff

Use AI Core–enabled SDKs Libraries and SDKs to handle retry logic. The SDKs provide configurable retry behavior. Use them instead of calling the APIs directly.

If you do not use the SDKs, implement retry logic as follows:

> **Note**
>
> Do not retry immediately after a `429` response. Use exponential backoff with jitter to prevent thundering-herd behavior.

1. When you receive a `429` response, wait for an initial interval (for example, 1 second).
2. If the response includes a `Retry-After` header, use its value as the wait interval.
3. After each failed retry, increase the wait time exponentially and add a small random jitter.
4. Limit retries by using a maximum cap (for example, 5 retries or 60 seconds total wait time).

Spread Requests Over Time

Avoid sending large request bursts. Distribute requests evenly to stay within limits.

Cache Responses

Reuse responses for repeated prompts to reduce API calls.

Use Model Fallbacks

Route requests to alternative models if the primary model is rate-limited.

Isolate Workloads

Assign different applications to separate resource groups to prevent shared quota exhaustion.

Request a Rate Limit Increase

Request a quota increase if mitigation strategies do not meet your workload requirements.

You can use the following API endpoints to check your quota limits and request updates to them.

Ensure that you've the following headers set:

Header	Value
Authorization	Bearer $AUTH_TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. This URL can also be set as an environment variable.

Check Current Limits

Send a GET request to the endpoint `$AI_API_URL/v2/admin/quota/model`

By default, you receive a **tenant-level** rate limit, and the quota applies to all resource groups.

For example:

```
curl --location "$AI_API_URL/v2/admin/quota/model" \
--header "Authorization: Bearer $AUTH_TOKEN"
```

Output Code

```
[
  {
    "resourceType": "model",
    "quotaDetails": [
      {
        "unit": "requestPerMinute",
        "limit": 78,
        "descriptors": {
           "modelName": "gpt-5",
           "modelVersion": "*",
           "providerName": "azure-openai"
        }
      },
      //... other models
    ]
  }
]
```

If a quota is requested for a specific resource group, it appears as part of the response. For **resource-group-level** limits, include the `AI-Resource-Group` header.

For example:

```
curl --location "$AI_API_URL/v2/admin/quota/model" \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header "Authorization: Bearer $AUTH_TOKEN"
```

Submit a Request

You can request a rate limit update by sending a POST request to the endpoint `$AI_API_URL/v2/admin/quota/requests`. Include the details of your rate limit update in your request.

> **Note**
>
> A successful submission of a rate-limit increase request returns a `requestId`. Requests go through an approval process.

Tenant level Request

Populate the following request with your values:

```
curl --location "$AI_API_URL/v2/admin/quota/requests" \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "resourceType": "model",
  "quotaDetails": {
    "unit": "requestPerMinute",
    "limit": <new limit>,
    "descriptors": {
      "modelName": "<modelName>",
      "providerName": "<model provider executableId>"
    }
  },
  "reason": "<Provide a reason for quota update request>"
}'
```

> **Output Code**
>
> ```
> {
> "requestId": "5ed1d56f-3d0c-45a4-aa55-1b97020fa714",
> "message": "The request has been submitted for approval."
> }
> ```

Resource Group Level requests

To request a rate limit update for a particular resource-group, include the resource group in your header and include `scope: resource_group` in your request.

> **Note**
>
> A successful submission of a rate-limit increase request returns a `requestId`. Requests go through an approval process.

For example:

```
curl --location "$AI_API_URL/v2/admin/quota/requests" \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
  "resourceType": "model",
  "scope": "resource_group",
  "quotaDetails": {
    "unit": "requestPerMinute",
    "limit": <new limit>,
    "descriptors": {
      "modelName": "<modelName>",
      "providerName": "<model provider executableId>"
    }
  },
  "reason": "<Provide a reason for quota update request>"
}'
```

Check Request Status

You can check the status of your rate limits by sending a GET request to the endpoint `$AI_API_URL/v2/admin/quota/requests`.

For example:

```
curl --location "$AI_API_URL/v2/admin/quota/requests" \
--header "Authorization: Bearer $AUTH_TOKEN"
```

To follow up on a request, create a support ticket on component CA-ML-AIC. For more information, see Support Process .

1.6.7 Inference Observability

Inference observability stores the payloads of inferences to foundation models or orchestration. The payload can be retrieved later for analytical purposes.

Key features include:

- You can add labels to be able to filter inferences.
- You can store feedback to a given inference after the inference has been recorded in inference observability.

To store the inference request, response, and feedback, you'll need an object store that was registered using an object store secret. For more information, see Register an Object Store for Inference Observability [page 776]

⚠ Restriction

Only S3 object stores are supported by inference observability.

Alternatively, you can opt to store only metadata and labels. In this case, no object store is required.

Metadata is additional information about the inference such as:

- Model name
- Model version
- Input tokens
- Output tokens
- Latency

Inferences (including metadata) are only stored if you explicitly declare in the inference headers that the request should be recorded.

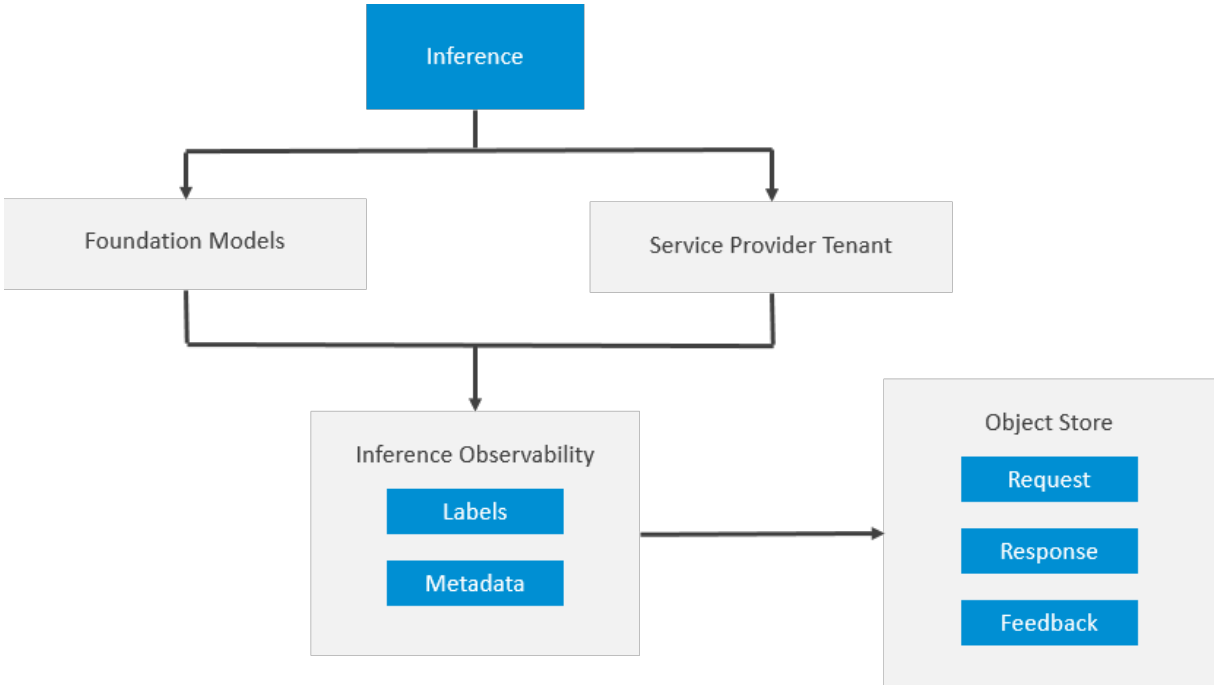
ⓘ Note

Request, response, and feedback are stored without data masking or additional processing in the object store. It is your responsibility to ensure that personal data is only recorded with appropriate consent.

⚠ Caution

The feedback, request and response are not sanitized throughout the inference observability workflow. It is the your responsibility to sanitize the input and to guard against injection attacks.

Inference Observability Architecture and Workflow



APIs are documented on the SAP Business Accelerator Hub. For more information, see REST API.

The costs for inference record storage are detailed in 3720903.

Inference Observability Workflow

- To store the inference request, response, and feedback, you'll need to register an object store for observability. For more information, see Register an Object Store for Inference Observability [page 776].
- You specify in the headers of your requests that you want the request to be recorded. For more information, see Record an Inference in Inference Observability [page 777].
- You can add labels or feedback to an inference record. For more information, see Adding to an Inference Record [page 779]
- You can retrieve inference records for your own analysis. For more information, see Retrieving an Inference Record [page 781]
- You can retrieve feedback associated with your inference records for your own analysis. For more information, see Retrieving Feedback [page 784]
- You can remove labels, feedback and metadata from an inference record. For more information, see Deleting Inference Record Data [page 785]

1.6.7.1 Prerequisites

1.6.7.1.1 Register an Object Store for Inference Observability

Connect SAP AI Core to a cloud object store and manage access using an object store secret. The connected storage stores your dataset, models, and other cache files of the Metaflow Library for.

Context

Your cloud storage credentials are managed using secrets. Secrets are a means of allowing and controlling connections across directories and tools, without compromising your credentials.

To store the inference request, response, and feedback, you'll need an object store that has been registered using an object store secret.

> ⚠ Restriction
>
> Only S3 object stores are supported by inference observability.

> ⚠ Caution
>
> You're responsible for the rotation of your access credentials and certificates of SAP AI Core within BTP according to regional policy.

Procedure

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Register the details of your object store secret using the endpoint `/v2/admin/objectStoreSecrets`. All `<data>` fields are required.

Sample Code

```
curl --location --request POST "$AI_API/admin/objectStoreSecrets" \
--header "Authorization: Bearer {{access_token}}" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: {{resource-group}}' \
--data-raw '{
  "name": "default",
  "type": "S3",
  "bucket": "<S3 bucket name>",
  "endpoint": "<S3 end point>",
  "pathPrefix": "<A path prefix that follows the bucket name>",
  "region": "<S3 region>",
  "data": {
    "AWS_ACCESS_KEY_ID": "<AWS access key ID>",
    "AWS_SECRET_ACCESS_KEY": "<AWS secret access key>"
  }
}'
```

Results

Successful responses return code **202** and include a success message.

1.6.7.2 Record an Inference in Inference Observability

Prerequisites

- You're using the extended service plan. For more information, see Service Plans.
- If you want to store your request, response, and feedback data: you've registered an object store secret. For more information, see Register an Object Store for Inference Observability [page 776].

- You have a running orchestration or model deployment in the target resource group. For more information, see Create a Deployment for Orchestration [page 703] or Foundation Models [page 190].

Procedure

To access generative AI models, send a request to an inference endpoint. For more information, see Orchestration Workflow V2 [page 382] or Consume Models [page 198].

To record your inference, you must include the following headers in your request:

Header	Value
`ai-inference-observability-persistence-mode`	Choose from the following: • `full`: stores the request and response payload, including the metadata. • `metadata`: metadata only. **Note** If you want to store feedback, you have to set `ai-inference-observability-persistence-mode` headers to `full`.
`ai-object-store-secret-name`	Required for requests with `full` persistence mode. Not required for `metadata` persistence mode. The object store secret name where your request and response payloads should be saved.
`ai-inference-observability-labels`	Optional Extra labels that will be attached to the metadata for later filtering. The following conditions apply: • Maximum of 16 labels • Comma-separated list of key-value pairs. • The key must use prefix `ext.ai.sap.com` • Maximum length of key and value: 64 characters. • Only uppercase and lowercase ASCII letters, numbers and the characters “-”, “.”, and “ /” are permitted. Example value: `ext.ai.sap.com/env=prod,ext.ai.sap.com/team=ml`

Note

Only inferences sent to orchestration or foundation models are recorded.

Results

Requests sent with `ai-inference` headers automatically record either the metadata or request, response payload, and metadata, depending on the selected mode.

The response includes an inference ID that can be used to access inference observability.

Next Steps

You can use your inference ID to:

- Store additional labels and feedback
- Retrieve the metadata and the request, response, and feedback (if stored).

1.6.7.3 Adding to an Inference Record

Prerequisites

- You've made a request to an LLM or orchestration, and included inference recording in your headers. You know the ID of the inference record that you want to access. For more information, see Record an Inference in Inference Observability [page 777].

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Adding Labels

Send a POST request to the endpoint: `$AI_API/observability/inferences/{{inferenceId}}/labels`. Include the ID of the inference that you want to update in your request. Include your labels as key-value pairs in the body.

> ⚠ Restriction
>
> Only new labels can be added. If you try to add a label with an existing key, the request will be rejected. Label keys must have the prefix `ext.ai.sap.com`.

Sample Code

```
curl --request POST \
  --url $AI_API/observability/inferences/{{inferenceId}}/labels \
  --header 'ai-resource-group: <resource_group>' \
  --header 'authorization: Bearer $TOKEN' \
  --header 'content-encoding: application/json' \
  --header 'content-type: application/json' \
  --data '[
  {
    "key": "ext.ai.sap.com/<key>",
    "value": "<value>"
  }
```

Result

Successful requests return a message confirming that the label has been stored with the inference record.

Output Code

```
{
  "message": "inference label stored."
}
```

Adding Feedback

Send a POST request to the endpoint: `$AI_API/observability/inferences/{{inferenceId}}/feedback`. Include the ID of the inference that you want to update in your request. Include your feedback in the body. The feedback can be any object.

Note

POST requests always add feedback. Existing feedback isn't overwritten.

Output Code

```
curl --request POST \
  --url $AI_API/observability/inferences/{{inferenceId}}/feedback \
  --header 'ai-resource-group: <resource_group>' \
  --header 'authorization: Bearer $TOKEN' \
  --header 'content-encoding: application/json' \
  --header 'content-type: application/json' \
  --data '[{ "content": {"stars": 5}}]'
```

Result

Successful requests rerun a message confirming that the feedback has been stored with the inference record.

Output Code

```
{
  "message": "feedback stored."
```

```
}
```

1.6.7.4 Retrieving an Inference Record

You can retrieve inference records individually by ID, or by retrieving multiple records associated with a predefined label.

Prerequisites

- You've made a request to an LLM or orchestration, and included inference recording in your headers. For more information, see Record an Inference in Inference Observability [page 777].
- You know the ID of the inference record that you want to access, or you've included labels in your record. For more information, see Record an Inference in Inference Observability [page 777] and Adding to an Inference Record [page 779].

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Retrieve an Inference Record by ID

Send a GET request to the endpoint: `$AI_API/observability/inferences/{{inferenceId}}`. Include the ID of the inference that you want to retrieve in your request.

Sample Code

```
curl --request GET \
  --url $AI_API/observability/inferences/{{inferenceId}} \
  --header 'ai-resource-group: <resource_group>' \
  --header 'authorization: Bearer $TOKEN' \
  --header 'content-type: application/json'
```

Result

For responses with `ai-inference-observability-persistence-mode` set to `full`: the response contains metadata, the request and response, and any labels or feedback added to the inference.

Output Code


```
{
  "id": "4ba60b4a-30ee-9070-9671-3fbca68ed085",
  "source": "llmaccess",
  "resourceGroup": "default",
  "deploymentId": "d64b3328cf209947",
  "modelName": "gpt-4o",
  "modelVersion": "2024-08-06",
  "streaming": true,
  "responseHttpCode": 200,
  "inputTokens": 21,
  "outputTokens": 15,
  "latency": 637.341,
  "createdAt": "2026-03-11T11:48:24.3133655Z",
  "objectStoreSecretName": "inference-observability-store",
  "request": "{\"url\": \"http://rg-openai-exploration-
us.openai.azure.com/openai/deployments/gpt-4o-2024-08-06/chat/completions?api-
version=2025-03-01-preview\", \"headers\": {\"Content-Type\": \"application/
json\", \"api-key\": \"7f32c2cf9bf645e3b5cc0df2943b35bc\"}, \"body\": \"{\\
\"messages\\\": [{\\\"role\\\": \\\"system\\\", \\\"content\\\": \\\"You are
a helpful assistant.\\\"}, {\\\"role\\\": \\\"user\\\", \\\"content\\\": \\
\"Tell me a joke\\\"}], \\\"temperature\\\": 1, \\\"stream\\\": true}\"}",
  "response": "{\"choices\":
[],\"created\":0,\"id\":\"\",\"model\":\"\",\"object\":\"\",\"prompt_filter_re
sults\":[{\"prompt_index\":0,\"content_filter_results\":{\"hate\":
{\"filtered\":false,\"severity\":\"safe\"},\"self_harm\":
{\"filtered\":false,\"severity\":\"safe\"},\"sexual\":
{\"filtered\":false,\"severity\":\"safe\"},\"violence\":
{\"filtered\":false,\"severity\":\"safe\"}}}]},
[ … ]
}
```


Note

If the associated object store secret isn't valid or can't be retrieved, only the metadata will be returned.

Retrieve Inference Records Using Labels

You can retrieve inferences by label key only, or using label key-value pairs.

Send a GET request to the endpoint: `$AI_API/observability/inferences?labelSelector=ext.ai.sap.com/<key>=<value>`. Include the key or key-value pair that you want to retrieve in your request.

The following example shows a request for `key`: `<medium>` and `value`: `<mobile>`:

Sample Code

```
curl --request GET \
  --url $AI_API/observability/inferences?labelSelector=ext.ai.sap.com/
medium=Mobile \
  --header 'ai-resource-group: <resource_group>' \
  --header 'authorization: Bearer $TOKEN' \
  --header 'content-type: application/json'
```

Result

The result includes a list of all matching inference records, and all associated labels and feedback:

Output Code


```
{
  "count": 10,
  "resources": [
    {
      "id": "4ba60b4a-30ee-9070-9671-3fbca68ed085",
      "labels": [
        {
          "key": "ext.ai.sap.com/use-case",
          "value": "Accounting"
        },
        {
          "key": "ext.ai.sap.com/medium",
          "value": "Mobile"
        }
      ],
      "source": "llmaccess",
      "resourceGroup": "default",
      "deploymentId": "d64b3328cf209947",
      "modelName": "gpt-4o",
      "modelVersion": "2024-08-06",
      "streaming": true,
      "responseHttpCode": 200,
      "inputTokens": 21,
      "outputTokens": 15,
      "latency": 637.341,
      "createdAt": "2026-03-11T11:48:24.3133655Z",
      "objectStoreSecretName": "default",
      "feedback": [
        {
          "content": {
            "stars": 5
          }
        }
      ]
      "request": "{\"url\": \"http://rg-openai-exploration-
us.openai.azure.com/openai/deployments/gpt-4o-2024-08-06/chat/completions?api-
version=2025-03-01-preview\", \"headers\": {\"Content-Type\": \"application/
json\", \"api-key\": \"7f32c2cf9bf645e3b5cc0df2943b35bc\"}, \"body\": \"{\\
\"messages\\\": [{\\\"role\\\": \\\"system\\\", \\\"content\\\": \\\"You are
a helpful assistant.\\\"}, {\\\"role\\\": \\\"user\\\", \\\"content\\\": \\
\"Tell me a joke\\\"}], \\\"temperature\\\": 1, \\\"stream\\\": true}\"}",
      "response": "{\"choices\":
[],\"created\":0,\"id\":\"\",\"model\":\"\",\"object\":\"\",\"prompt_filter_re
sults\":[{\"prompt_index\":0,\"content_filter_results\":{\"hate\":
{\"filtered\":false,\"severity\":\"safe\"},\"self_harm\":
{\"filtered\":false,\"severity\":\"safe\"},\"sexual\":
{\"filtered\":false,\"severity\":\"safe\"},\"violence\":
{\"filtered\":false,\"severity\":\"safe\"}}}]},
[….]
```

1.6.7.5 Retrieving Feedback

You can retrieve all stored feedback of an inference record.

Prerequisites

- You've made a request to an LLM or orchestration, and included inference recording in your headers, and you know the ID of the inference record that you want to access. For more information, see Record an Inference in Inference Observability [page 777].
- You've added feedback to your record. For more information, see Adding to an Inference Record [page 779].

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Procedure

Send a GET request to the endpoint: `$AI_API/observability/inferences/{{inferenceId}}/feedback`. Include the ID of the inference that you want to retrieve in your request.

Sample Code

```
curl --request GET \
  --url $AI_API/observability/inferences/{{inferenceId}}/feedback \
  --header 'ai-resource-group: <resource_group>' \
  --header 'authorization: Bearer $TOKEN' \
  --header 'content-encoding: application/json'
```

Results

The result is a list of all stored feedback objects.

Output Code

```
{
  "count": 1,
  "resources": [
    {
```

```
      "content": {
        "stars": 5
      }
    ]
}
```

1.6.7.6 Deleting Inference Record Data

Deletion of inference record data includes all of metadata and labels, but not the request, response, or feedback.

Prerequisites

- You've made a request to an LLM or orchestration, and included inference recording in your headers. For more information, see Record an Inference in Inference Observability [page 777].
- You know the ID of the inference record that you want to access, or you've included labels in your record. For more information, see Record an Inference in Inference Observability [page 777] and Adding to an Inference Record [page 779].

Make sure that you've set the following headers:

Header	Value
Authorization	Bearer $TOKEN
AI-Resource-Group	The resource group used in the activation steps
$AI_API_URL	The base URL of your SAP AI Core environment. You can set the URL as an environment variable.

Context

> **Note**
>
> If no selection criteria are provided, all inference record data for the specific resource group will be deleted.

The API supports bulk deletion of all objects that fit your selection criteria.

You can specify a time-frame and label selection criteria.

> **Caution**
>
> Data stored in the S3 data store such as request, response, and feedback, is not deleted. It is your responsibility to delete the data from the S3 store.

Procedure

1. Send a POST request to the endpoint: `$AI_API/observability/inferencesDeleteRequests`.
   Include your selection criteria in the body.
   The following example shows a request for deletion of data associated with `key`: `<medium>` and `value`: `<mobile>`, created within a specific time-frame:

   Sample Code

   ```
   curl --request POST \
     --url $AI_API/observability/inferencesDeleteRequests \
     --header 'ai-resource-group: <resource_group>' \
     --header 'authorization: Bearer $TOKEN' \
     --header 'content-encoding: application/json' \
     --header 'content-type: application/json' \
     --data '  {
       "startTime": "2026-03-16T10:22:23.012Z",
       "endTime": "2026-03-17T10:22:23.012Z",
       "labelSelectors": [
         {
           "key": "ext.ai.sap.com/medium",
           "value": "mobile"
         }
       ]
     }'
   ```

   The response includes the ID of the delete request.

   Output Code

   ```
   {
     "id": "74d5227a-fb5e-4fae-82cb-3b4f77b7a6b9"
   }
   ```

2. Use the ID of the delete request to check the status of the delete request.

Sample Code

```
curl --request GET \
  --url $AI_API/observability/inferencesDeleteRequests/
{{inferencesDeleteRequestId}} \
  --header 'ai-resource-group: <resource_group>' \
  --header 'authorization: Bearer $TOKEN' \
  --header 'content-encoding: application/json'
```

Results

Responses include a success message and a confirmation of the criteria used for deletion.

Output Code

```
{
  "status": "COMPLETED",
  "filters": {
```

```
    "labelSelectors": [
      {
        "key": "ext.ai.sap.com/medium",
        "value": "mobile"
      }
    ],
    "startTime": "2026-03-16T10:22:23.012Z",
    "endTime": "2026-03-17T10:22:23.012Z"
  },
  "submittedAt": "2026-03-17T10:37:25.2043074Z",
  "completedAt": "2026-03-17T10:40:06.5120414Z"
}
```

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering an SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

© 2026 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see https://www.sap.com/about/legal/trademark.html for additional trademark information and notices.