

PUBLIC  
2025-10-22

## SAP AI Core

# Content

<b>1</b>	<b>What Is SAP AI Core? . . . . .</b>	<b>6</b>
1.1	Metering and Pricing for SAP AI Core. . . . .	9
	Metering and Pricing for Generative AI. . . . .	10
	Metering and Pricing for Predictive AI. . . . .	14
1.2	Choose an Instance. . . . .	16
	Service Usage Reporting. . . . .	18
<b>2</b>	<b>What's New for SAP AI Core . . . . .</b>	<b>19</b>
<b>3</b>	<b>Concepts. . . . .</b>	<b>39</b>
3.1	SAP AI Core Overview . . . . .	46
	SAP AI Core Systems Overview. . . . .	47
	AI API Overview. . . . .	50
	Resource Groups. . . . .	54
3.2	Generative AI Hub in SAP AI Core Overview. . . . .	55
<b>4</b>	<b>Initial Setup. . . . .</b>	<b>57</b>
4.1	Enabling the Service in Cloud Foundry. . . . .	57
	Create a Subaccount. . . . .	58
	Enable Cloud Foundry. . . . .	60
	Create a Space. . . . .	61
	Add a Service Plan. . . . .	62
	Create a Service Instance. . . . .	68
	Create a Service Key. . . . .	72
	Use a Service Key. . . . .	75
4.2	Enabling the Service in the Kyma Environment. . . . .	78
<b>5</b>	<b>SAP AI Core Starter Tutorials. . . . .</b>	<b>79</b>
<b>6</b>	<b>Administration. . . . .</b>	<b>80</b>
6.1	Manage Your Git Repository. . . . .	80
	Add a Git Repository. . . . .	80
	Edit a Git Repository. . . . .	83
	Delete a Git Repository. . . . .	84
6.2	Manage Applications. . . . .	86
	Create an Application. . . . .	86
	List Applications. . . . .	89
	Edit an Application. . . . .	89

Delete an Application.	90
<b>6.3 Manage Resource Groups.</b>	<b>90</b>
Resource Group Level Resources	91
Create a Resource Group	92
Edit a Resource Group	93
Delete a Resource Group	95
<b>6.4 Manage Object Store Secrets.</b>	<b>96</b>
Register an Object Store Secret	96
Edit an Object Store Secret	101
Delete an Object Store Secret	106
<b>6.5 Manage Docker Registry Secrets.</b>	<b>107</b>
Register Your Docker Registry Secret	107
Edit a Docker Registry Secret	109
Delete a Docker Registry Secret	111
<b>6.6 Manage Generic Secrets.</b>	<b>111</b>
Create a Generic Secret	111
Get Generic Secrets	116
Update a Generic Secret	119
Delete a Generic Secret	122
Consume Generic Secrets in Executions or Deployments	123
<b>7 Generative AI Hub in SAP AI Core.</b>	<b>125</b>
<b>7.1 Quick Start.</b>	<b>126</b>
Get an Auth Token	127
Get Your Orchestration Deployment URL	128
Consume Models with the Harmonized API	129
Next Steps	131
<b>7.2 Accessing Generative AI Models Through Global Scenarios.</b>	<b>131</b>
Orchestration	132
Foundation Models	393
<b>7.3 Supported Models.</b>	<b>418</b>
<b>7.4 Additional Concepts.</b>	<b>421</b>
Create a Deployment for Orchestration	421
Prompt Registry	423
Rate Limit Management	447
<b>8 Predictive AI.</b>	<b>450</b>
<b>8.1 ML Operations.</b>	<b>450</b>
Connect Your Data	450
Train Your Model	460
Use Your Model	509
<b>8.2 Metrics.</b>	<b>555</b>

Metrics Tracking with AI API . . . . .	555
<b>9 APIs and API Extensions . . . . .</b>	<b>572</b>
<b>10 Libraries and SDKs . . . . .</b>	<b>573</b>
<b>11 Content Packages . . . . .</b>	<b>575</b>
<b>12 Tutorials . . . . .</b>	<b>576</b>
<b>13 Advanced Features . . . . .</b>	<b>577</b>
13.1 AI Content as a Service . . . . .	577
Service Custom Resource . . . . .	579
Getting Started as a Service Provider . . . . .	581
Metering . . . . .	583
Offboarding . . . . .	583
Shared Resource Group . . . . .	584
<b>14 Security . . . . .</b>	<b>587</b>
14.1 Security Features of Data, Data Flow, and Processes . . . . .	587
14.2 Encryption in Transit . . . . .	587
14.3 Authentication and Administration . . . . .	588
14.4 Docker Images . . . . .	588
14.5 AI Content Security . . . . .	588
14.6 Kubernetes Security . . . . .	589
14.7 Configuration Data and Secrets . . . . .	590
14.8 Output Encoding . . . . .	590
14.9 Multitenancy . . . . .	591
14.10 Auditing and Logging Information . . . . .	592
14.11 Data Protection and Privacy . . . . .	594
Data Storage and Processing . . . . .	595
Change Logging and Read-Access Logging . . . . .	595
Consent . . . . .	595
Deletion . . . . .	596
Security and Customer Data Protection . . . . .	596
<b>15 Accessibility Features in SAP AI Core . . . . .</b>	<b>597</b>
<b>16 Monitoring and Troubleshooting . . . . .</b>	<b>598</b>
16.1 Troubleshooting . . . . .	598
Repository . . . . .	599
Configuration . . . . .	600
Artifacts . . . . .	602
Application . . . . .	604
Execution . . . . .	609

Docker	611
Deployment	612
Miscellaneous	613
<b>17 Support Process</b>	<b>616</b>
<b>18 Service Offboarding</b>	<b>617</b>

# 1 What Is SAP AI Core?

Learn more about the SAP AI Core service on SAP Business Technology Platform (SAP BTP). Build a platform for your artificial intelligence solutions.

SAP AI Core is a service within the SAP Business Technology Platform. It's designed to manage the execution and operations of AI assets in a standardized, scalable, and hyperscaler-agnostic manner. It seamlessly integrates with SAP solutions, allowing any AI function to be easily implemented using open-source frameworks. SAP AI Core supports full lifecycle management of AI scenarios. Users can access generative AI capabilities and prompt lifecycle management through the generative AI hub.

SAP AI Core lets you make data-driven decisions confidently and efficiently, tailored to address business challenges. It handles large volumes of data and offers scalable machine learning capabilities to automate tasks like triaging customer feedback or tickets and performing classification tasks. SAP AI Core includes preconfigured SAP solutions and supports open-source machine learning frameworks. It integrates with Argo Workflow and KServe, and can be embedded into other applications.

SAP AI Core allows you to experiment with and utilize natural language prompts with a variety of generative AI models in the generative AI hub.

## → Tip

The English version of this guide is open for contributions and feedback using GitHub. This allows you to get in contact with responsible authors of SAP Help Portal pages and the development team to discuss documentation-related issues. To contribute to this guide, or to provide feedback, choose the corresponding option on SAP Help Portal:

-  [Feedback](#)  [Create issue](#): Provide feedback about a documentation page. This option opens an issue on GitHub.
-  [Feedback](#)  [Edit page](#): Contribute to a documentation page. This option opens a pull request on GitHub.

You need a GitHub account to use these options.

More information:

- [Contribution Guidelines](#)
- [Introduction Video](#) 
- [Introduction Blog Post](#) 

## Features

### Generative AI hub

Choose from a selection of generative AI models for prompt experimentation and prompt lifecycle management.

<b>Execute pipelines</b>	Execute pipelines as a batch job, for example, to preprocess or train your models, or perform batch inference.
<b>Serve inference requests</b>	Deploy a trained machine learning model as a Web service to serve inference requests of trained models with high performance.
<b>Manage the AI scenario lifecycle</b>	Manage your ML artifacts and workflows, such as model training, metrics tracking, data, models, and model deployments via a uniform API lifecycle.
<b>Benefit from multitenancy support</b>	Use this service in tenant-aware applications. Implement multi-tenant services to segregate your AI assets and executions to isolate your tenants within SAP AI Core.
<b>Integrate your cloud infrastructure</b>	Register your Docker registry, synchronize your AI content from your git repository, and register your object store for training data and trained models. Productize your AI content and expose it as a service to consumers in the SAP BTP marketplace.

## Environment

This service is available in the following environments:

- Cloud Foundry
- Kyma
- Kubernetes

## Multitenancy

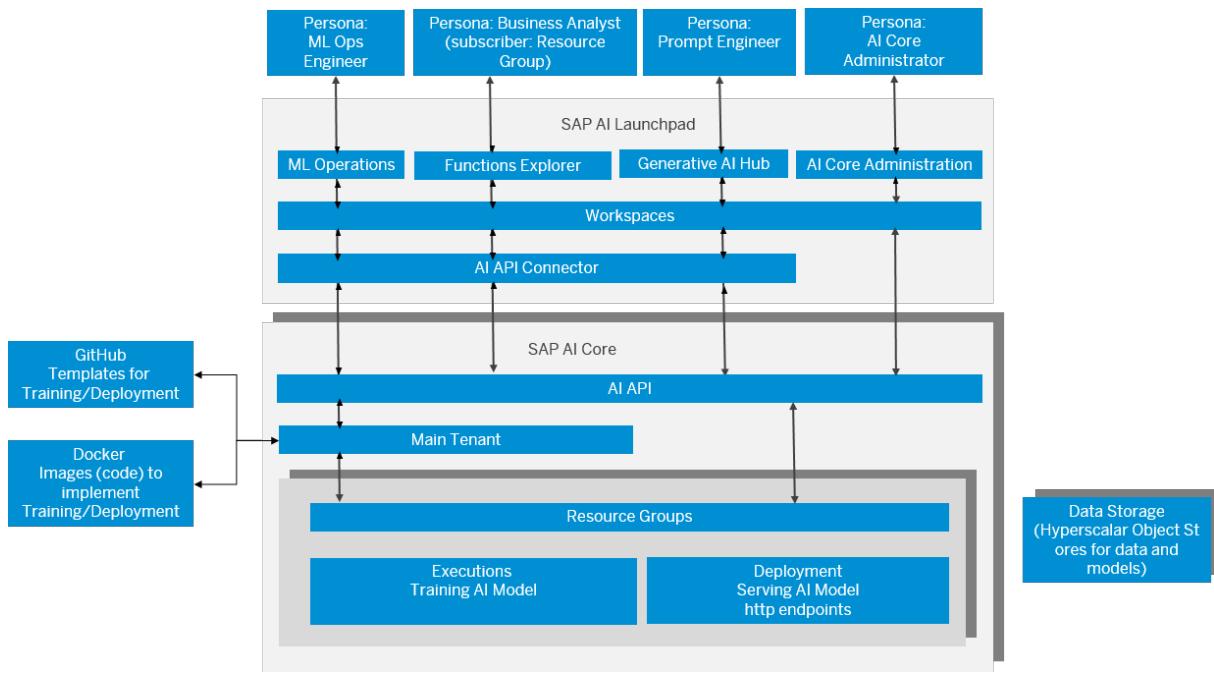
This service supports multitenancy. It can be used in tenant-aware applications. For more information, see [Multitenancy \[page 591\]](#).

## Architectural Overview

The SAP AI Core service works in conjunction with SAP AI Launchpad and the AI API. The key components are SAP AI Core, SAP AI Launchpad, and the AI API.

- **SAP AI Core** provides an engine that lets you run AI workflows and model serving workloads.
- **SAP AI Launchpad** manages a number of AI runtimes. It allows various user groups to access and manage their AI scenarios.
- **AI API** provides a standard way of managing the AI scenario lifecycle on different runtimes, regardless of whether they're provided on SAP technology (such as SAP S/4HANA) or on partner technology (such as Amazon Web Services). When the AI API is deployed on runtimes other than SAP AI Core, the runtimes have to provide a runtime adapter.

[Architectural Overview \[page 8\]](#) shows these three main components in an overview architecture diagram.



Architectural Overview

## Tools

SAP AI Core connects with various internal and external tools. You interact with different repositories, systems, and objects. Some of these objects come from SAP, while others must be provided by you. This setup enables enhanced control through authorizations, and supports continuous integration and continuous deployment (CI/CD). The following table lists the key repositories, systems, and objects:

What	Why
AI API	For managing your artifacts and workflows (such as training scripts, data, models, and model servers) across multiple runtimes
<p><b>ⓘ Note</b></p> <p>The AI API can also be used to integrate other machine learning platforms, engines, or runtimes into the AI ecosystem.</p>	
Argo Workflows	A container native, workflow engine for Kubernetes.
Docker repo	For custom Docker images referenced in the templates
Git repo	For storing training and serving workflows and templates
Hyperscaler storage	For storage of input and output artifacts, such as training data and models (for example, SAP BTP Object Store Service)
KServing (K)	For optimized deployments of machine learning models. Deployment templates use KServe notation.
Kubernetes (K8s)	The K8s cluster orchestrates and scales the pods, which are used in AI pipelines. Resource group isolation is based on a K8s namespace.

What	Why
SAP AI Launchpad	SAP AI Launchpad is a multitenant software as a service (SaaS) application on SAP Business Technology Platform (SAP BTP). Customers and partners can use SAP AI Launchpad to manage AI use cases (scenarios) across multiple instances of AI runtimes (such as SAP AI Core). SAP AI Launchpad also provides generative AI capabilities via the Generative AI Hub.

## Related Information

[Generative AI Hub in SAP AI Core \[page 125\]](#)

[AI API Overview \[page 50\]](#)

[SAP AI Launchpad](#)

## 1.1 Metering and Pricing for SAP AI Core

SAP AI Core provides a scalable infrastructure for AI model management, with usage-based pricing that lets you pay only for the resources you use.

The service offers various plans, including a free, standard, and extended plan. If you want to access all capabilities, we recommend the extended plan.

We offer the following model options:

- Generative AI with SAP-hosted, SAP-managed, and remote foundation models, and pipelines such as grounding
- Predictive AI with your custom AI models

You have the flexibility to select the model options that best suit your needs.

Custom AI models incur compute, storage, and baseline costs. The system automatically applies baseline charges every hour, up to a maximum of 730 hours.

If you use foundation models and grounding, the platform waives compute, storage, and baseline costs. Instead, charges accrue based on the number of tokens used by the foundation models.

## More Information

For more information about the potential costs associated with using foundation models and grounding in SAP AI Core, including metering information, example pricing calculations, and typical consumption patterns, see [Metering and Pricing for Generative AI \[page 10\]](#).

For more information about the potential costs associated with using custom AI models in SAP AI Core, including metering information, example pricing calculations, and typical consumption patterns, see [Metering and Pricing for Predictive AI \[page 14\]](#).

## Consumption Information

To see detailed consumption information in SAP BTP cockpit, open your global account and click [Usage](#) from the navigation panel. Filter by AI Core and click Export. The report contains generative AI hub model consumption under [Applications](#) and resource group level consumption under [instance](#).

For more information, see [Monitoring Usage and Consumption Costs in Your Global Account in SAP BTP](#).

### 1.1.1 Metering and Pricing for Generative AI

The use of generative AI in SAP AI Core is measured using tokens, which are converted into capacity units based on input and output volume. Billing is calculated according to these capacity units and varies by model type and usage pattern.

#### ⓘ Note

The generative AI hub is available only as part of the extended service plan.

### Metering for Generative AI

Use of large language models (LLMs) is metered in tokens. Tokens are the fundamental building blocks of text that a language model processes. They represent single characters, parts of words, entire words, or even punctuation marks. They're the unit into which the LLM breaks down text input and output.

It's important to distinguish between input and output tokens. Input tokens refer to the text provided in the prompt, while output tokens represent the response generated by the LLM. The ratio between input and output tokens varies depending on the prompt and the use case. For example, in summarization tasks, the ratio of input to output tokens could be 3:1 or even 4:1. In contrast, tasks like generating explanations or drafting longer pieces of text could have an inverse ratio, with output tokens significantly outnumbering input tokens.

The number of tokens required depends on the content and complexity of your prompt and the specific task assigned to the LLM. Token consumption varies with the capabilities of the LLM being used, with more advanced models typically requiring and processing a higher number of tokens.

Consumption of generative AI in SAP AI Core is measured in GenAI tokens. GenAI tokens are a virtual unit used to represent usage across different models. The actual number of processed tokens (input and output) that corresponds to one GenAI token varies by model.

Generative AI resource requirements vary based on the use case. Available resources are shown in the table:

Resource Type	SAP AI Core Resources	Unit of Measure (UoM)
Provisioning of foundation models	Access to (LLMs) and other generative AI capabilities	Tokens
Baseline	Grounding Service	Gigabyte day

## Pricing for Generative AI

The billing metric for SAP BTP is referred to as “capacity units” (CUs). Capacity units are calculated using conversion factors applied to GenAI tokens.

In general, output tokens tend to be slightly more expensive than input tokens.

For more information about the supported models, including the conversion rates between model tokens and GenAI tokens, see SAP Note [3437766](#).

Here's an example estimation, using fictitious values:

### Example

This example contains:

- A retrieval-augmented generation (RAG) system handling 25,000 requests in a single month results in the following costs:
- Language model use: The total of 25,000 requests, each with 3,500 input tokens and 300 output tokens.
- Grounding service: The RAG pipeline retrieves and processes context using the following orchestration modules:
  - Storage in Grounding (for storing relevant documents)
  - Retrieval in Grounding (for fetching context during inference)
  - Content Filtering
  - Data Masking (for privacy and compliance)

The following orchestration services are essential to support the grounding step and introduce additional capacity usage beyond the core LLM calls.

- The Grounding storage module stores contextual documents in the vector engine.
- The Grounding retrieval module retrieves context documents during inference.
- Content Filtering via Meta Llama Guard helps to ensure content safety and relevance.
- Data Masking, handled by SAP Data Privacy Integration, applies masking at the API level.

Each module uses its own metric (storage size, text blocks, API calls) multiplied by monthly usage, then converted to capacity units to estimate resource consumption and cost.

### Calculate the calls incurred through LLM calls:

This example uses a model with an input GenAI token conversion rate of 0.00112 per 1,000 input tokens. The output GenAI token conversion rate is 0.00320 per 1,000 output tokens.

- Calculate Total GenAI tokens

Formula: Total GenAI tokens =  $(x/1000 \times \text{conversion rate per 1,000 input tokens}) + (y/1000 \times \text{conversion rate per 1000 output tokens})$

In this formula, "x" is the number of input tokens and "y" is the number of output tokens. The conversion rates per 1,000 input and output tokens are given constants.

This example includes 3,500 input tokens and 300 output tokens. The conversion rate is 0.00112 per 1,000 input tokens and 0.00320 per 1,000 output tokens.

Values: Total GenAI Tokens =  $(3500/1000 \times 0.00112) + (300/1000 \times 0.00320) = 0.00488$

- Calculate capacity units

Formula: capacity units = Total GenAI Tokens  $\times 1.90385$

In this formula, "Total GenAI Tokens" is the value calculated in the previous step and 1.90385 the capacity unit value, given as a constant.

Values: capacity units =  $0.00488 \times 1.90385 = 0.00929$

- Calculate Total Usage for handling 25,000 requests

Formula: Total capacity units = Requests  $\times$  capacity units per request

In this formula, "Requests" is the number of requests to be handled (in this case, 25,000) and "capacity units per request" is the value calculated in the previous step.

Values: Total capacity units =  $25,000 \times 0.00929 = \$232.25$

### Calculate Grounding Module Costs:

The RAG architecture needs various supporting services to handle contextual data for the language model. These services use resources measured by specific metrics, which are converted into capacity units based on usage volume and internal conversion factors

Each module uses its own metric (storage size, text blocks, API calls) multiplied by monthly usage. The values are converted to capacity units to quantify resource consumption and estimate cost.

Module	Service	Metric	Metric Value	Occurrences / Month	Capacity Units / Month
Grounding	Storage	Storage in GB / Day	1	30	6.2 CU
Grounding	Retrieval	Text Blocks	1	25.000	30 CU
Content Filter	Meta Llama Guard	Text Blocks	5	25.000	84 CU
Data Masking	SAP Data Privacy Integration	API Calls	1	25.000	66.1 CU
Total					186.3 CU

Total consumption =  $232.25 + 186.3 = 418.55$  CU

## Images

For image use with Mistral Small Instruct, images are converted into tokens in batches of 14\*14 pixels. The formula for the number of image tokens is: (ResolutionX/14) \* (ResolutionY/14). Images with a resolution higher than 1540\*1540 are downscaled, while maintaining their aspect ratio.

### Example

This example uses fictitious values.

Given an image of resolution 720\*512, the number of GenAI tokens consumed through the image is:

$$\text{GenAI Image tokens} = (720/14)*(512/14) = 1,880$$

The calculation for image tokens also includes additional sequence start and end tokens, as well as tokens for image breaks.

## Typical Consumption Patterns

The following table displays API request counts for productive IT use cases based on the consumption patterns "small", "medium", and "large". It also specifies the average distribution of input and output tokens, which can help you estimate potential costs for consuming foundation models.

Use Case	Input Tokens	Output Tokens	Tokens Occurrences / Request per Month: Small	Tokens Occurrences / Request per Month: Medium	Tokens Occurrences / Request per Month: Large
RAG Chat	3500	300	1000	25k	300k
Basic Chat	500	100	1000	30k	300k
Summarization	5000	300	1000	25k	300k
Classification	3800	10	1000	5k	50k
Generation	500	3500	1000	2k	50k

The generative AI hub provides additional modules through the Orchestration API, including:

- Grounding
- Content Filtering
- Data Masking

Charges associated with the use of other SAP AI Core components can also apply. For more information, see [Metering and Pricing for Predictive AI \[page 14\]](#).

### → Recommendation

For an estimate of projected costs, use the [SAP AI Core Cost Calculator](#).

## 1.1.2 Metering and Pricing for Predictive AI

### Metering for Predictive AI

Custom AI workloads involve diverse resource types, each with varying performance and efficiency levels. Measuring key resources like compute power, storage, and baseline costs is crucial. Non-billable units of measure (UoM) are used to represent the consumption of these resources.

For example, compute-heavy AI models on high-performance instances consume more resources per hour than smaller, less demanding workloads. This results in proportionally higher costs. Pricing reflects true resource consumption.

Custom AI development requirements vary significantly based on the use case. A range of compute infrastructure resources is available, differing in CPU cores, memory (GB), and GPU presence for GPU-powered workloads like model training. Available resources are shown in the table.

#### ⓘ Note

Infrastructure specifications depend on the chosen hyperscaler. For more information, see [Choose an Instance \[page 462\]](#).

Resource Type	SAP AI Core Resources	Unit of Measure (UoM)
Compute	Starter Instance (min. 2.5 GB, 1 CPU core)	Node hour
	Basic Instance (min. 10 GB, 3 CPU cores)	
	Basic.8x Instance (min. 115 GB, 31 CPU cores)	
	Infer-S Instance (min. 10 GB, 3 CPU cores, standard GPU)	
	Infer-M Instance (min. 25 GB, 7 CPU cores, standard GPU)	
	Infer-L Instance (min. 55 GB, 15 CPU cores, standard GPU)	
Storage	Train-L Instance (min. 47 GB, 5 CPU cores, advanced GPU)	
	Standard SSD Volume	Gigabyte hour

Resource Type	SAP AI Core Resources	Unit of Measure (UoM)
Baseline	Fixed cluster resources (instances, storage, SAP HANA, Kubernetes, DevOps, support, and so on)	Tenant hour

Custom AI models can incur compute, storage, and baseline costs. You have the flexibility to select the components that best suit your needs. If you use compute or storage resources, the system automatically applies baseline charges every hour, up to a maximum of 730 hours.

### ⓘ Note

You can process data in parallel by using multiple nodes. Costs are calculated in node hours. For example, processing data for 1 hour on 2 nodes equals 2 node hours. However, it incurs only 1 hour of baseline costs.

## Pricing for Predictive AI

The billing metric for SAP BTP is referred to as “capacity units”. Capacity units are calculated using conversion factors applied to compute, storage, and baseline intermediary UoM values. This allows you to calculate a specific price for the infrastructure used based on your compute and storage requirements.

Here's an example calculation, using fictitious values:

### ⚡ Example

Training a small model on a GPU node and serving it on a CPU node later in the same month results in the following costs:

- Baseline: Total node hours of compute instances across the full month
- Storage: Temporary high-speed storage (for example, SSD) used during training and inference to stream or copy data
- Compute: Compute consumption based on node hours and chosen compute instances across the full month

Resource Type	Unit of Measure	Capacity Unit (CU)	Value	Fictional Usage	Conversion to CU
Compute	node hour	0.3900		300 node hours in Basic Instance	$300 * 0.39 = 117$ CU
Compute	node hour	1.552		100 node hours in Infer-M Instance	$100 * 1.552 = 155.2$ CU
Storage	gigabyte hour	0.0003		10.000 gigabyte hours	$0.0003 * 5,000 = 1.5$ CU

Resource Type	Unit of Measure	Capacity Unit (CU)		Conversion to CU
		Value	Fictional Usage	
Baseline	tenant hour	1.2241	400 baseline tenant hours	1.2241*400 = 489.64 CUs

Infrastructure specifications are combined minimum values; actual numbers vary by deployed hyperscaler and may be higher.

The CUs can be converted to monetary values in the BTP Discovery Center Estimator. For more information, see [BTP Discovery Center Estimator](#).

#### → Recommendation

For an estimate of projected costs, use the [SAP AI Core Cost Calculator](#).

## Typical Consumption Patterns

Depending on your infrastructure and your consumption pattern (S, M or L), you need on average the following node hours for compute, storage, and baseline hours.

T Shirt									
Size	Starter	Basic	Basic8x	Infer.S	Infer.M	Infer.L	Train.L	Storage	Baseline
S	50	30	0	300	0	0	50	0	430
M	0	900	0	160	1880	1440	120	0	730
L	0	0	0	800	10000	0	2300	0	730

## 1.2 Choose an Instance

You can configure SAP AI Core to use different infrastructure instances for different tasks, based on demand. SAP AI Core provides several preconfigured infrastructure bundles called “resource plans” and “instance types” for this purpose.

## Context

You must choose at least one instance. If you select a resource plan, an instance type isn't required and vice versa.

Resource plans and instance types are used to select instances in workflow and serving templates. Different steps of a workflow can have different instances assigned.

In general, if your workload needs GPU acceleration, use one of the GPU-enabled instances. Otherwise, choose a plan based on the anticipated CPU and memory need of your workloads.

Within SAP AI Core, the instances are selected via the `ai.sap.com/resourcePlan` and `ai.sap.com/instanceType` labels at pod level. It maps the selected instance and takes a string value, which is the ID of the instance.

There are limits to the default disk storage size for all these nodes. Datasets that are loaded to the nodes consume disk space. If you have large data sets (larger than 30 GB), or have large models, you may have to increase the disk size. To do so, use the persistent volume claim in Argo Workflows to specify the required disk size (see [Volumes](#) ).

## Instance Types

### ⚠ Restriction

Instance types are only available as part of the “extended” service plan. For more information on instances that can be used with the “standard” service plan.

Within SAP AI Core, the instances are selected via the `ai.sap.com/instanceType` label in your workflow templates. It maps the selected resource and takes a string value, which is the ID of the instance

For information about available instance types and their IDs, see SAP Note [3660109](#), or send a GET request to the following endpoint:

```
{apiurl}/v2/admin/resources/instanceType
```

For example:

### ↔ Sample Code

```
curl GET "$AI_API_URL/v2/admin/resources/instanceType" \
--header "Authorization: Bearer $TOKEN" \
```

## Resource Plans

Within SAP AI Core, the instances are selected via the `ai.sap.com/resourcePlan` label in your workflow templates. It maps the selected resource and takes a string value, which is the ID of the instance.

## Resource Plan Specifications for AWS

Resource Plan IDs	GPUs	CPU Cores	Memory GBs	Code to Allocate instances in Workflow Templates
Starter	-	1	3	ai.sap.com/resourcePlan:starter
Basic	-	3	11	ai.sap.com/resourcePlan:basic
Basic-8x	-	31	116	ai.sap.com/resourcePlan:basic.8x
Train-L	1 V100	7	55	ai.sap.com/resourcePlan:train.l
Infer-S	1 T4	3	10	ai.sap.com/resourcePlan:infer.s
Infer-M	1 T4	7	26	ai.sap.com/resourcePlan:infer.m
Infer-L	1 T4	15	58	ai.sap.com/resourcePlan:infer.l

### ⓘ Note

For the free service plan, only the Starter resource plan is available. Specifying other plans results in an error. For the standard service plan, all resource plans are available.

## Service Usage Reporting

Usage consumption of services is reported in the SAP BTP cockpit on the [Overview](#) page for your global account and on the [Overview](#) and [Usage Analytics](#) pages of your subaccount. The usage report lists usage in billable measures and non-billable measures. Your final monthly bill is based on the billable measure only. Non-billable measures are displayed for reporting purposes only.

## 2 What's New for SAP AI Core

Tech nical Envi- ron- men- t		Title		Description	Action	Life- cy- cle	Type	Line of busi- ness	Busi- ness Proc	Mod- ular	Lat- est Rev- ision	Avail- able as of
SAP AI Core	Cloud Foundry	Generative AI hub		New models are supported, including Perplexity Sonar and Sonar Pro.  For more information, see SAP Note <a href="#">3437766</a> .	Info only	Gen- eral Avail- abil- ity	An- noun ce- men t	Tech nology	Not applic- able	SAP Business Technology Platform	2025 -10-2 0	2025 -10-2 0
SAP AI Core	Cloud Foundry	Generative AI hub		SAP Document Management is supported as a repository type for the orchestration grounding module.  For more information, see <a href="#">Grounding Generic Secrets for SAP Document Management Service [page 282]</a> .	Info only	Gen- eral Avail- abil- ity	An- noun ce- men t	Tech nology	Not applic- able	SAP Business Technology Platform	2025 -09- 01	2025 -09- 01
SAP AI Core	Cloud Foundry	Generative AI hub		Orchestration model configurations have timeout and max tries parameters.  For more information, see <a href="#">Templating [page 344]</a> .	Info only	Gen- eral Avail- abil- ity	An- noun ce- men t	Tech nology	Not applic- able	SAP Business Technology Platform	2025 -09- 01	2025 -09- 01
SAP AI Core	Cloud Foundry	Generative AI hub		New models are supported, including GPT 5, GPT 5 Mini, and GPT 5 Nano.  For more information, see SAP Note <a href="#">3437766</a> .	Info only	Gen- eral Avail- abil- ity	An- noun ce- men t	Tech nology	Not applic- able	SAP Business Technology Platform	2025 -08-1 8	2025 -08-1 8
SAP AI Core	Cloud Foundry	Generative AI hub		New models are supported, including Gemini Embeddings.  For more information, see SAP Note <a href="#">3437766</a> .	Info only	Gen- eral Avail- abil- ity	An- noun ce- men t	Tech nology	Not applic- able	SAP Business Technology Platform	2025 -08- 06	2025 -08- 06

Tech						Modular			Lat-		
nical	Envi-	Com	ron-	po-	men	Ac-	Life-	Line	Busi	est	Avail
ment	t	Title	Description	tion	cle	Type	ness	ness	Proc	Rev-	able
SAP	Clou	Generative	This initial orchestration endpoint is deprecated and is scheduled for decommissioning on October 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 SAP Note <a href="#">Orchestration Workflow V2 [page 259]</a> and <a href="#">3634540</a> .	Info only	Dep-re-cate d	An-noun ce-men t	Technolo-gy	Not-appli-ca-ble	SAP Business Technology Platform	2025-07-21	2025-07-21
SAP AI Core	Cloud Foundry	AI hub	New models are supported, including Nova Premier from AWS Bedrock.  For more information, see SAP Note <a href="#">3437766</a> .	Info only	General Avail-ability	An-noun ce-men t	Technolo-gy	Not-appli-ca-ble	SAP Business Technology Platform	2025-07-21	2025-07-21
SAP AI Core	Cloud Foundry	Generative AI hub	Prompt shields are available as part of input filtering with Azure Content Safety. Prompt shields detect and mitigate prompt attacks, such as prompts designed to bypass safety mechanisms or override previous instruction.  For more information, see <a href="#">Harm categories in Azure AI Content Safety</a> and <a href="#">Input Filtering</a> .	Info only	General Avail-ability	An-noun ce-men t	Technolo-gy	Not-appli-ca-ble	SAP Business Technology Platform	2025-06-23	2025-06-23

Tech				Modular					Lat-		
nical	Envi-	Com	ron-	Life-	Line	Busi	est	Avail			
po-	men	pon-	men	Ac-	cy-	Busi	Proc	Rev-	able		
nen-	Title	Description		Action	Cycle	Type	Ness	Product			
SAP AI Core	Cloud Foundry	Generative AI hub	GPT-4 versions 0613 and turbo-2024-04-09 and gpt-4-32k version 0613 have been deprecated.	Info only	Dep-re-rate d	An-noun ce-men t	Tech nology	Not applicable	SAP Business Technology Platform	2025-06-02	2025-06-02
			It's recommended that users of these models migrate to GPT-4.1 as a replacement.								
			For more information, see SAP Note <a href="#">3437766</a> .								
SAP AI Core	Cloud Foundry	Generative AI hub	Gemini-1.5-pro and gemini-1.5-flash have been deprecated.	Info only	Dep-re-rate d	An-noun ce-men t	Tech nology	Not applicable	SAP Business Technology Platform	2025-06-02	2025-06-02
			It's recommended that users of these models migrate to gemini-2.0-flash, or gemini-2.0-flash-lite as replacements.								
			For more information, see SAP Note <a href="#">3437766</a> .								
SAP AI Core	Cloud Foundry	Generative AI hub	Mistralai--mixtral08x7b-v01 has been deprecated.	Info only	Dep-re-rate d	An-noun ce-men t	Tech nology	Not applicable	SAP Business Technology Platform	2025-06-02	2025-06-02
			It's recommended that users of this model migrate to mistralai--mistral-small-instruct as a replacement.								
			For more information, see SAP Note <a href="#">3437766</a> .								
SAP AI Core	Cloud Foundry	Generative AI hub	New models are supported, including OpenAI GPT, o3, o4-mini, 4.1, 4.1-mini, 4.1-nano and Mistral small instruct.	Info only	General Avail-ability	An-noun ce-men t	Tech nology	Not applicable	SAP Business Technology Platform	2025-05-19	2025-05-19
			For more information, see SAP Note <a href="#">3437766</a> .								

Tech				Modular					Lat-		
nical	Envi-	Com	ron-	Life-	Line	Busi		est	Avail		
po-	men	pon-	men	Ac-	cy-	of	ness	Proc	Rev-	able	
nen-	t	Title	Description	tion	cle	Type	ness	ess	Product	sion	as of
SAP AI Core	Cloud Foundry	Generative AI hub	An additional module has been added to orchestration.  The translation module translates text, and can be configured for input and output text.  The input translation module helps improve answer quality when the configured model performs better when input is provided in a specific language, for example English.  For more information, see <a href="#">Input Translation [page 235]</a> and <a href="#">Output Translation [page 256]</a> .	Info only	General Availability	Answer men-	Technology	Not applicable	SAP Business Technology Platform	2025-04-24	2025-04-24
SAP AI Core	Cloud Foundry	SAP BTP Cockpit	A detailed consumption report is available in SAP BTP for Generative AI hub as the part of SAP AI Core service. You can see the consumption specific to an LLM model and Orchestration service, and consumption for each resource group.  For more information, see <a href="#">Metering and Pricing for Generative AI [page 10]</a> .	Info only	General Availability	Answer men-	Technology	Not applicable	SAP Business Technology Platform	2025-04-14	2025-04-14
SAP AI Core	Cloud Foundry	Generative AI hub	New models are supported, including Anthropic Claude 3.7 sonnet, GCP Vertex AI Gemini 2.0-flash and flash-lite, NVIDIA Llama 3.2 nv embedqa 1b, and OpenAI GPT 4o version 2024-11-20.  For more information, see SAP Note <a href="#">3437766</a> .	Info only	General Availability	Answer men-	Technology	Not applicable	SAP Business Technology Platform	2025-04-14	2025-04-14

Tech nical Envir- Com- pon- ment						Life- acy- tion			Line of Busi- ness	Busi- ness	Mod- ular	Lat- est Rev- sion	Avail- able as of
AI Core	Cloud Foundry	Title	Description	Type	Ac- tive	Line of Business	Busi- ness	Proc- ess	Product				
SAP AI Core	Cloud Foundry	Generative AI hub	We now support the Anthropic models Claude Opus 4 and Claude Sonnet 4.  For more information, see SAP Note <a href="#">3437766</a> .	Info only	General Availability	Anonymous men	Technology	Not applicable	SAP Business Technology Platform	2025-06-23	2025-06-23		
SAP AI Core	Cloud Foundry	Generative AI hub	More endpoints have been added to the Pipeline API, so you can retrieve information about the documents within a pipeline, including processing status.	Info only	General Availability	Anonymous men	Technology	Not applicable	SAP Business Technology Platform	2025-02-03	2025-02-03		
SAP AI Core	Cloud Foundry	Generative AI hub	Selected models from Aleph Alpha are supported.  For more information, see SAP Note <a href="#">3437766</a> .	Info only	General Availability	Anonymous men	Technology	Not applicable	SAP Business Technology Platform	2025-02-03	2025-02-03		
SAP AI Core	Cloud Foundry	Generative AI hub	Additional selected models and model versions from Open AI are supported.  For more information, see <a href="#">3437766</a> .	Info only	General Availability	Anonymous men	Technology	Not applicable	SAP Business Technology Platform	2025-03-03	2025-03-03		
SAP AI Core	Cloud Foundry	Generative AI hub	You can now configure your orchestration workflow so that Llama Guard 3 filters input and output content. Llama Guard 3 provides several content categories for filtering content.  Azure Content Safety is still supported.  For more information, see <a href="#">Input Filtering</a> [page 247] and <a href="#">Output Filtering</a> [page 252].	Info only	General Availability	Anonymous men	Technology	Not applicable	SAP Business Technology Platform	2025-01-09	2025-01-09		

Tech nical Envir- Compon- ment						Modular			Line of Busi- ness			Lat- est Revi- sion		
		Title	Description	Action	Lifecycle	Type	Business	Process	Product	Available	As of			
SAP AI Core	Cloud Foundry	Generative AI hub	<p>Prompt registry</p> <p>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.</p> <p>For more information, see <a href="#">Prompt Registry [page 423]</a>.</p>	Info only	General Availability	Anonymous men-tability	Technology	Not applicable	SAP Business Technology Platform	2024-12-02	2024-12-02			
SAP AI Core	Cloud Foundry	Generative AI hub	<p>Retrieval and vector APIs added.</p> <p>The vector API can be used for managing collections and documents in the Vector database.</p> <p>The retrieval API searches data repositories and returns the relevant chunks for the user query.</p> <p>For more information, see <a href="#">Preparing Data Using the Vector API [page 315]</a> and <a href="#">Retrieval API [page 332]</a>.</p>	Info only	General Availability	Anonymous men-tability	Technology	Not applicable	SAP Business Technology Platform	2024-12-02	2024-12-02			
SAP AI Core	Cloud Foundry	Generative AI hub	<p>Additional selected models and model versions from Anthropic via AWS Bedrock are supported.</p> <p>For more information, see <a href="#">Foundation Models [page 393]</a> and SAP Note 3437766.</p>	Info only	General Availability	Anonymous men-tability	Technology	Not applicable	SAP Business Technology Platform	2024-12-02	2024-12-02			
SAP AI Core	Cloud Foundry	Generative AI hub	<p>Additional selected models and model versions from Open AI are supported.</p> <p>For more information, see SAP Note 3437766.</p>	Info only	General Availability	Anonymous men-tability	Technology	Not applicable	SAP Business Technology Platform	2024-12-02	2024-12-02			

Tech nical Envir- Compon- men- t				Mod- ular				Lat- est Rev- sion		
				Life- acy- tion	Line of Busi- ness	Busi- ness	Pro- cess	Product	Avail- able as of	
SAP AI Core	Cloud Foundry	Generative AI hub	Additional selected model versions from Vertex AI are supported.  For more information, see SAP Note <a href="#">3437766</a> .	Info only	General Availability	Named Entity Recognition	Technology	SAP Business Technology Platform	2024-12-02	2024-12-02
SAP AI Core	Cloud Foundry	Generic Secrets	Generic secrets are now available at the tenant-wide level.  For more information, see <a href="#">Create a Generic Secret [page 111]</a> .	Info only	General Availability	Named Entity Recognition	Technology	SAP Business Technology Platform	2024-12-02	2024-12-02
SAP AI Core	Cloud Foundry	Generative AI hub	Pipeline API added.  This pipeline segments data into chunks and generates embeddings, which are multidimensional representations of textual information. The embeddings are stored in a vector database.  For more information, see <a href="#">Create a Document Grounding Pipeline Using the Pipelines API (without Metadata) [page 293]</a> .	Info only	General Availability	Named Entity Recognition	Technology	SAP Business Technology Platform	2024-11-04	2024-11-04
SAP AI Core	Cloud Foundry	Generative AI hub	A new module has been added to the orchestration system. The grounding module enhances AI processes by integrating external data that is contextually relevant, domain-specific, or real-time. This additional data complements the natural language processing abilities of pre-trained models, which are typically based on general information.  For more information, see <a href="#">Grounding [page 265]</a> .	Info only	General Availability	Named Entity Recognition	Technology	SAP Business Technology Platform	2024-11-04	2024-11-04

Tech nical Envi- ron- men- t		Title		Description	Ac- tion	Life- cy- cle	Type	Line of Busi- ness	Busi- ness	Mod- ular	Lat- est	Avail- able
Com- po- nen- t	AI	Cloud Foundry	AI hub					Business	Proc- ess	Product	Rev- ision	As of
SAP AI Core	Cloud Foundry	Generative AI hub		Additional selected models from IBM are supported.  For more information, see SAP Note <a href="#">3437766</a> .	Info only	Gen- eral Avail- abil- ity	An- noun ce- men- t	Tech nology	Not applic- able	SAP Business Technology Platform	2024 -10-0 7	2024 -10-0 7
SAP AI Core	Cloud Foundry	Generative AI hub		Additional selected models from Mistral AI are supported.  For more information, see SAP Note <a href="#">3437766</a> .	Info only	Gen- eral Avail- abil- ity	An- noun ce- men- t	Tech nology	Not applic- able	SAP Business Technology Platform	2024 -10-0 7	2024 -10-0 7
SAP AI Core	Cloud Foundry	Generative AI hub		For selected models, response streaming is supported in the generative AI hub orchestration service.  For more information, see <a href="#">Streaming [page 384]</a> .	Info only	Gen- eral Avail- abil- ity	An- noun ce- men- t	Tech nology	Not applic- able	SAP Business Technology Platform	2024 -10-0 7	2024 -10-0 7
SAP AI Core	Cloud Foundry	Generative AI hub		When creating a deployment for orchestration, you can restrict the choice of models using an allow or disallow list. You can use this to implement internal standards, for example where only certain LLMs are approved for use.  For more information, see <a href="#">Create a Deployment for Orchestration [page 421]</a> .	Info only	Gen- eral Avail- abil- ity	An- noun ce- men- t	Tech nology	Not applic- able	SAP Business Technology Platform	2024 -09-1 6	2024 -09-1 6
SAP AI Core	Cloud Foundry	Generative AI hub		Additional module added to orchestration.  The data masking module anonymizes or pseudonymizes personally identifiable information from input.  For more information, see <a href="#">Orchestration Workflow V1 (Deprecated) [page 132]</a> .	Info only	Gen- eral Avail- abil- ity	An- noun ce- men- t	Tech nology	Not applic- able	SAP Business Technology Platform	2024 -09-1 6	2024 -09-1 6

Tech						Modular			Lat-			
nical	Envi-	Com	ron-	po-	men	Ac-	Life-	Line	Busi	est	Avail	
ment	t	Title	Description			cycl	of	ness	Proc	Rev-	able	
						Action	Type	Business	Process	Product	Revision	
											as of	
SAP AI Core	Cloud Foundry	Generative AI hub	Additional selected models from Meta are supported.  For more information, see SAP Note <a href="#">3437766</a> .			Info only	General Availability	Anonymous men-t	Technology	SAP Business Technology Platform	2024-09-16	2024-09-16
SAP AI Core	Cloud Foundry	Generative AI hub	tiuae--falcon-40b-instruct is deprecated and is unavailable from 2024-09-01. Users should choose another model.  For more information, see SAP Note <a href="#">3437766</a> .			Info only	Deprecated	Changed	Technology	SAP Business Technology Platform	2024-09-01	2024-09-01
SAP AI Core	Cloud Foundry	Generative AI hub	Additional selected models from OpenAI are supported via Azure.  For more information, see SAP Note <a href="#">3437766</a> .			Info only	General Availability	Anonymous men-t	Technology	SAP Business Technology Platform	2024-09-02	2024-09-02
SAP AI Core	Cloud Foundry	Generative AI hub	Additional selected models from Anthropic are supported via AWS Bedrock.  For more information, see SAP Note <a href="#">3437766</a> .			Info only	General Availability	Anonymous men-t	Technology	SAP Business Technology Platform	2024-05-20	2024-05-20
SAP AI Core	Cloud Foundry	Generative AI hub	Orchestration combines content generation with a set of functions that are often required in business AI scenarios.  Functions include templating, which lets you compose a prompt with placeholders that are filled during inference, and content filtering, which lets you restrict the type of content that is passed to and received from a generative AI model.			Info only	General Availability	Anonymous men-t	Technology	SAP Business Technology Platform	2024-08-05	2024-08-05

Tech nical Envir- Com- pon- ment				Mod- ular				Line of Busi- ness		Lat- est Rev- sion		Avail- able as of	
		Title	Description	Action	Life- cy- cle	Type	Business	Process	Product				
SAP AI Core	Cloud Foundry	Generative AI hub	Additional selected models from GCP Vertex AI are supported. For more information, see SAP Note <a href="#">3437766</a> .	Info only	General Availability	An-noun-men-t	Tech-nology	Not applicable	SAP Business Technology Platform	2024-07-08	2024-07-08		
SAP AI Core	Cloud Foundry	Resource limits per tenant	The maximum number of applications, git repository secrets, docker registry secrets, workflow templates and deployment templates is limited at tenant level to 50 per resource. If you reach this limit, you will receive an error message. You can free up space by deleting resources.	Info only	General Availability	An-noun-men-t	Tech-nology	Not applicable	SAP Business Technology Platform	2024-06-24	2024-06-24		
SAP AI Core	Cloud Foundry	Generative AI hub	Additional selected models from OpenAI are supported via Azure. For more information, see SAP Note <a href="#">3437766</a> .	Info only	General Availability	An-noun-men-t	Tech-nology	Not applicable	SAP Business Technology Platform	2024-06-24	2024-06-24		
SAP AI Core	Cloud Foundry	Generative AI hub	Selected models from Amazon and Anthropic are supported via AWS Bedrock. For more information, see SAP Note <a href="#">3437766</a> .	Info only	General Availability	An-noun-men-t	Tech-nology	Not applicable	SAP Business Technology Platform	2024-06-03	2024-06-03		
SAP AI Core	Cloud Foundry	Generative AI hub	Selected models from Meta are supported. For more information, see SAP Note <a href="#">3437766</a> .	Info only	General Availability	An-noun-men-t	Tech-nology	Not applicable	SAP Business Technology Platform	2024-06-03	2024-06-03		
SAP AI Core	Cloud Foundry	Generative AI hub	Selected models from Mistral AI are supported. For more information, see SAP Note <a href="#">3437766</a> .	Info only	General Availability	An-noun-men-t	Tech-nology	Not applicable	SAP Business Technology Platform	2024-05-20	2024-05-20		

Tech				Modular					Lat-		
nical	Envi-	Com	ron-	Life-	Line	Busi		est	Avail		
po-	men	pon-	men	Ac-	cy-	of	ness	Proc	Rev-	able	
nen-	t	Title	Description	Action	Cycle	Type	Business	Process	Product	Revision	As of
SAP AI Core	Cloud Foundry	Deployments	When deployments are submitted, configurations are checked for errors, synchronously.  For more information, see <a href="#">Deploy Models [page 531]</a> .	Info only	General Availability	Anonymen	Technology	Not applicable	SAP Business Technology Platform	2024-04-08	2024-05-20
SAP AI Core	Cloud Foundry	Generative AI hub	Selected models from GCP Vertex AI are supported.  For more information, see SAP Note <a href="#">3437766</a> .	Info only	General Availability	Anonymen	Technology	Not applicable	SAP Business Technology Platform	2024-04-08	2024-04-08
SAP AI Core	Cloud Foundry	Resource groups limit per tenant	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 <a href="#">Delete a Resource Group [page 95]</a> .	Info only	General Availability	Anonymen	Technology	Not applicable	SAP Business Technology Platform	2024-03-08	2024-03-08
SAP AI Core	Cloud Foundry	Authentication	Authentication through X.509 certificates is supported.  For more information, see <a href="#">Create a Service Key [page 72]</a> and <a href="#">Use a Service Key [page 75]</a> .	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2024-02-18	2024-02-18
SAP AI Core	Cloud Foundry	generative AI hub SDK	The generative AI hub SDK is now available.  For more information, see <a href="#">generative AI hub SDK</a> .	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2024-02-18	2024-02-18

Tech nical Enviro- Compon- ment				Life- acy- cycle		Line of busi- ness	Busi- ness	Mod- ular	Lat- est Rev- sion		Avail- able as of
Title		Description		Action	Type	New	Tech	Not	Product		
SAP AI Core	Cloud Foundry	Generative AI hub	The generative AI hub includes prompt experimentation, prompt management, and administrative tools. Prompt experimentation includes creating and running natural language prompts with a choice of large language models and parameters. Prompt management includes saving prompts with collections, metadata in the form of tags and notes, versioning and deletion. For more information, see <a href="#">Generative AI Hub in SAP AI Core Overview [page 55]</a> .	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2023-12-20	2023-12-20
SAP AI Core	Cloud Foundry	Artifact signatures for artifact outputs from executions	Artifact signatures (hashes) can be generated and made available to other executions and deployments to verify the integrity of an artifact. For more information, see <a href="#">Using Artifact Signatures [page 482]</a> .	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2023-07-31	2023-07-31
SAP AI Core	Cloud Foundry	Template Generator	A wizard to generate workflow and serving templates in VS Code. Using user responses, it simplifies and automates the template writing process.  For more information, see <a href="#">SAP AI Core toolkit documentation</a> .	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2023-08-04	2023-08-04
SAP AI Core	Cloud Foundry	SAP AI Core Toolkit	SAP AI Core is available through the VS Code GUI, through the SAP AI Core toolkit extension.  For more information, see <a href="#">SAP AI Core toolkit documentation</a> .	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2023-08-04	2023-08-04

Tech nical Envir- Compon- ment						Life- Ac- tion	Line of Busi- ness	Busi- ness	Mod- ular	Lat- est Revi- sion	Avail- able as of
Title	Description	cle	Type	ness	Proc	Product					
SAP AI Core	Cloud Foundry	LLM Package	The content package for large language models for SAP AI Core simplifies the deployment of large language models with integrated and automated workflows.	Info only	General Availability	New	Techology	Not applicable	SAP Business Technology Platform	2023-05-31	2023-05-31
			For more information, see <a href="#">PyPi LLM</a> .								
SAP AI Core	Cloud Foundry	Dataset API	You can upload, download, and delete artifacts using the SAP AI Core Dataset API, when direct access to files in the object store is not possible or desirable. Currently, ****Using a Third-Party API Platform**** and curl interfaces are supported.	Info only	General Availability	New	Techology	Not applicable	SAP Business Technology Platform	2023-04-24	2023-05-02
SAP AI Core	Cloud Foundry	Metadata in Response to List Executables	When you use the endpoint to list executables, the response body now contains metadata about the parameters and artifacts. For parameters, the <code>description</code> and <code>default</code> values are returned. For artifacts, the <code>kind</code> , <code>description</code> , and <code>labels</code> can be added using annotations.	Info only	General Availability	Changed	Techology	Not applicable	SAP Business Technology Platform	2023-04-02	2023-04-02
			For more information, see <a href="#">List Executables [page 472]</a> , <a href="#">Workflow Templates [page 464]</a> , and <a href="#">Serving Templates [page 513]</a> .								

Tech nical Envir- Compon- ment				Life- Ac- tion		Line of Type	Busi- ness	Busi- ness	Mod- ular	Lat- est Revi- sion	Avail- able as of
				cle	cy-	ness	ess	Product			
SAP AI Core	Cloud Foundry	Contribute to Our Documentation	At SAP, we endeavor to make sure that our documentation works for you. If you feel that something is missing or that something doesn't quite hit the mark, you can now provide feedback and suggest changes directly from SAP Help Portal. You can do so in one of two ways: <ul style="list-style-type: none"> <li>Click <a href="#">Edit</a> in the toolbar to open the document in GitHub. There, you can suggest a change and submit a pull request for us to review.</li> <li>Click <a href="#">Feedback</a> in the toolbar to create a GitHub issue and tell us how we can improve the documentation for you.</li> </ul> For more information, including guidelines on how to contribute, see <a href="#">Open Documentation Initiative</a> .	Info only	General Availability	New	Techology	Not applicable	SAP Business Technology Platform	2023-03-13	2023-03-13
SAP AI Core	Cloud Foundry	Bulk PATCH Endpoint to STOP or DELETE	Executions and Deployments can now receive PATCH requests for bulk adjustments, provided bulkUpdates is enabled in the relevant template. For more information, see <a href="#">Workflow Templates [page 464]</a> and <a href="#">Serving Templates [page 513]</a> .	Info only	General Availability	New	Techology	Not applicable	SAP Business Technology Platform	2023-02-27	2023-02-09
SAP AI Core	Cloud Foundry	Sync Endpoint for ArgoCD	In addition to automatically syncing applications, you can request a sync manually by using an API endpoint.  For more information on syncing applications, see <a href="#">Create an Application [page 86]</a> .	Info only	General Availability	New	Techology	Not applicable	SAP Business Technology Platform	2023-02-27	2023-02-09

Tech nical Envir- Compon- ment						Life- acy- cycle			Line of Busi- ness		Busi- ness Proc- ess		Mod- ular		Lat- est Rev- ision		Avail- able as of	
		Title	Description	Action	Type	New	Tech	Not	Product									
SAP AI Core	Cloud Foundry	WebHDFS Artifacts	WebHDFS artifacts are now supported.  For more information about WebHDFS artifacts on SAP AI Launchpad, see <a href="#">Register an Object Store Secret [page 96]</a> .	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2023-02-27	2023-02-09							
SAP AI Core	Cloud Foundry	Periodic Scheduling	Executions can be run automatically, to a prepared schedule. For more information, see <a href="#">Training Schedules [page 503]</a> .	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2023-02-27	2023-02-09							
SAP AI Core	Cloud Foundry	Deployment Duration can be Limited	You can now use the <code>ttl</code> parameter to limit the duration of deployments to hours, days, or weeks.	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2022-11-20	2022-11-20							
SAP AI Core	Cloud Foundry	The YAML Files for Starter Tutorials Available Directly from GitHub	The relevant tutorial steps have been updated to include the link to the associated files. The YAML code can also be copied and pasted directly from the tutorials as before.	Info only	General Availability	Changed	Technology	Not applicable	SAP Business Technology Platform	2022-11-20	2022-11-20							
SAP AI Core	Cloud Foundry	Enhancements to the GET Deployments API Call Response	The GET Deployments response provides the number of min and max and running replicas, and the resource plan name. For more information, see <a href="#">the SAP AI Core API specification</a> .	Info only	General Availability	Changed	Technology	Not applicable	SAP Business Technology Platform	2022-10-18	2022-10-18							

Tech nical Envi- ron- men- t		Title		Description		Action	Life- cy- cle	Type	Line of busi- ness	Busi- ness	Mod- ular	Lat- est Rev- ision	Avail- able as of
Com- po- nent	AI	Cloud Foundry	Cloud Foundry	Title	Description								
SAP AI Core	Cloud Foundry	Azure Blob Storage	Supported	Azure Blob Storage	You can now register Azure Blob Storage secrets and use them for Model Serving.	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2022-10-18	2022-10-18
					For more information about registering secret, see <a href="#">Register an Object Store Secret [page 96]</a>								
SAP AI Core	Cloud Foundry	Free Tier Service Plan	Free Tier Service Plan	To try SAP AI Core for free, you can use a free tier service plan.	A free tier service plan can be easily upgraded to a standard plan, retaining your users and data.	Info only	General Availability	Announcement	Technology	Not applicable	SAP Business Technology Platform	2022-10-18	2022-10-18
					For more information, see <a href="#">Set Up the Free Plan [page 66]</a> .								
SAP AI Core	Cloud Foundry	Deployment Duration Variable	Deployment Duration Variable	Deployment Duration Variable	You can now limit the duration of a deployment by specifying the length of time that it should run for, in whole minutes hours or days.	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2022-10-04	2022-10-04
SAP AI Core	Cloud Foundry	Metrics Extension	Metrics Extension	Metrics Extension	If you have an AI API-enabled runtime, you can use the new "Metrics" extension to query which capabilities of the metrics endpoint are supported. You can specify the metrics capabilities to a very fine level of granularity, which lets you react in your client implementation accordingly.	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2022-09-19	2022-09-19
					For more information, see <a href="#">Metrics Extension</a> .								
SAP AI Core	Cloud Foundry	Git Repository Name	Git Repository Name	Git Repository Name	When you register your git repository, you no longer have to enter the repository name. The field has been removed. Repositories already registered with names assigned will still work.	Info only	General Availability	Announcement	Technology	Not applicable	SAP Business Technology Platform	2022-09-19	2022-09-19

Tech				Modular						Lat-			
nical Env-	Com ron-	po- men	ponent t	Title	Description	Ac- tion	Life- cy- cle	Line of Busi- ness	Busi- ness	Proc	est Rev- sion	Avail- able as of	
SAP AI Core	Cloud Foundry	API Deprecation and Decommissioning		API Deprecation and Decommissioning	API endpoints <code>POST /lm/configurations/{configurationId}/executions</code> , <code>POST /lm/configurations/{configurationId}/deployments</code> , <code>GET /lm/kpis</code> and <code>GET /analytics/resourceGroups</code> are deprecated. Please update your existing API calls. For timetables and more information, see SAP Note <a href="#">3239609</a> .	Re- quired	Dep- re- cated	Cha- nged	Tech- nology	Not ap- pli- ca- ble	SAP Business Technology Platform	2022-09-05	2022-09-05
SAP AI Core	Cloud Foundry	Serving Component Upgraded to KServe Version 0.7 for Security Compliance		Serving Component Upgraded to KServe Version 0.7 for Security Compliance	The upgrade means some changes to serving templates: the <code>spec.template.api</code> version changes from <code>v1beta1</code> to <code>serving.kubeflow.org/v1beta1</code> , the <code>spec.template.spec.predictor.containers.name</code> changes from <code>kfserving-container</code> to <code>kserve-container</code> . There is no support for <code>apiVersion: serving.kubeflow.org/v1alpha2</code> . Please update your existing serving templates by September 30, 2022.	Re- quired	Gen- eral Avail- abil- ity	Cha- nged	Tech- nology	Not ap- pli- ca- ble	SAP Business Technology Platform	2022-08-22	2022-08-22

Tech				Modular		Line of Business		Business Process		Last Revision		
Tech	Envi-	Com-	pon-	Life-	cy-	Type	ness	Proc	Product	Lat-	est	Avail-
nical	Envi-	Com-	ron-	Ac-	cle					Revi-	able	
Com-	ron-	pon-	men-	Life-	cy-	Type	ness	Proc	Product	Lat-	est	Avail-
ponent	t	Title	Description	Action	Cycle					Revision	as of	
SAP AI Core	Cloud Foundry	Object Stores	<p>SAP AI Core supports multiple hyperscaler object stores, such as Amazon S3, OSS, and HANA Data Lake (HDL).</p> <p>For more information about registering secret, see <a href="#">Register an Object Store Secret [page 96]</a></p>	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2022-06-30	2022-06-30	
SAP AI Core	Cloud Foundry	Enhanced Serving Template Parameters	<p>The <code>executables.ai.sap.com /cascade-update-deployments</code> parameter can be used in the serving template, to update associated deployments automatically. For more information, <a href="#">Serving Templates</a> and <a href="#">Change Serving Template and Update Deployments</a>.</p>	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2022-06-18	2022-06-18	
SAP AI Core	Cloud Foundry	Tracking Service Improvement	The performance of tracking is improved.	Info only	General Availability	Anonymous	Technology	Not applicable	SAP Business Technology Platform	2022-06-18	2022-06-18	
SAP AI Core	Cloud Foundry	Documentation for the AI api client SDK has moved to PyPi.org.	<p>Documentation for the AI api client SDK has moved to PyPi.org. For more information, <a href="#">Libraries and SDKs</a>.</p>	Info only	General Availability	Anonymous	Technology	Not applicable	SAP Business Technology Platform	2022-06-07	2022-06-07	
SAP AI Core	Cloud Foundry	Documentation for the sap-ai-sdk-core has moved to PyPi.org.	<p>Documentation for the sap-ai-sdk-core has moved to PyPi.org. For more information, <a href="#">Libraries and SDKs</a>.</p>	Info only	General Availability	Anonymous	Technology	Not applicable	SAP Business Technology Platform	2022-06-07	2022-06-07	

Tech nical Enviro- Compon- ment						Life- acy- cycle			Line of Busi- ness		Busi- ness		Mod- ular		Lat- est Rev- sion		Avail- able as of	
	Title	Description			Action	Type	New	Tech	Not	Product								
SAP AI Core	Cloud Foundry	Additional Query Parameter for Metrics	The \$select query parameter can be used to retrieve metric resource data, such as metrics or custom info selectively. For more information see, <a href="#">Querying Metric Data</a> .	Info only	General Availability	Changed	New	Technology	Not applicable	SAP Business Technology Platform	2022-05-05	2022-05-05	2022-05-05	2022-05-05	2022-05-05	2022-05-05		
SAP AI Core	Cloud Foundry	Horizontal Scaling for Resource Group Operator Enabled	The Resource Group Operator now allows horizontal scalability, improving performance.	Info only	General Availability	Changed	New	Technology	Not applicable	SAP Business Technology Platform	2022-04-09	2022-04-09	2022-04-09	2022-04-09	2022-04-09	2022-04-09		
SAP AI Core	Cloud Foundry	Resource Group Id Length Extended	The resource group Id length was limited to 10 characters, however longer Ids are now supported. Ids must be of length minimum: 3, maximum: 253. The first and last characters must be either a lower case letter, an upper case letter or, a number. Character entries from the second to penultimate, must be either a lower case letter, an upper case letter, a number, a full stop, or a hyphen. No other special characters are permitted.	Info only	General Availability	Changed	New	Technology	Not applicable	SAP Business Technology Platform	2022-04-09	2022-04-09	2022-04-09	2022-04-09	2022-04-09	2022-04-09		
SAP AI Core	Cloud Foundry	Generic Secret Support at Main Tenant Level	You can now store generic secrets at the main tenant level and use them with a generic service broker. For more information, see <a href="#">Register Generic Secrets</a> .	Info only	General Availability	Changed	New	Technology	Not applicable	SAP Business Technology Platform	2022-02-19	2022-02-19	2022-02-19	2022-02-19	2022-02-19	2022-02-19		

Tech				Modular				Lat-			
nical	Envi-	Com	ron-	Life-	Line	Busi		est	Avail	Rev-	able
po-	men	po-	men	Ac-	cy-	Busi	Proc				
nent	t	Title	Description	Ac-	cycl	Type	ness	ess	Product	sion	as of
SAP AI Core	Cloud Foundry	AI API meta Endpoint	The new meta API endpoint lets clients query the capabilities of a runtime engine that implements AI API. Querying the meta endpoint returns information that the client can use to trigger an appropriate response. For example, SAP AI Launchpad could enable or disable certain features based on the capability of the runtime engine implementation of AI API. For more information, see <a href="#">AI API Runtime Implementations</a> .	Info only	General Availability	New	Technology	Not applicable	SAP Business Technology Platform	2022-02-19	2022-02-19

# 3 Concepts

In this section, we'll explore some of the concepts surrounding both the SAP AI Core and SAP AI Launchpad services.

Term	Definition
AI API	An application programming interface that manages and organizes AI artifacts and workflows (such as training scripts, datasets, models, and model servers) across multiple AI runtimes (environments where AI models are executed, such as cloud platforms or edge devices) to facilitate the development, deployment, and monitoring of AI applications.
AI API connection	A link between SAP AI Launchpad and a runtime. This connection is established through the AI API. For more information, see <a href="#">About the AI API</a> .
AI scenario consumer	A user, either paying or non-paying, who utilizes an AI-powered software application, API, or platform for tasks such as automation, prediction, or decision-making. The consumer may have varying levels of technical expertise, ranging from non-technical end-users to experienced developers.
AI scenario producer	The developer or provider responsible for creating, maintaining, and updating an AI scenario. AI scenarios can be managed in three ways: <ul style="list-style-type: none"><li>• SAP-managed: Developed and maintained by SAP</li><li>• Partner-managed: Developed and maintained by SAP partners</li><li>• In-house: Developed and maintained by the organization using the AI scenario</li></ul>
AI service	A Software as a Service (SaaS) offering provided by SAP Business Technology Platform that enables users to leverage artificial intelligence capabilities for specific use cases. It is typically accessed through an AI API and can be used to expose and deploy AI scenarios as consumable services.
AI use case	A specific application of AI technology designed to generate business value by improving efficiency, reducing costs, or enhancing customer satisfaction. Examples include invoice matching in finance, product recommendations in e-commerce, product review classification in customer service, fraud detection in banking, and predictive maintenance in manufacturing.
artifact	Any tangible by-product produced or consumed by an execution or deployment. Artifacts can include data, files, binaries, libraries, packages, or other resources. They serve as inputs or outputs for various stages of the software lifecycle and are often versioned to track changes and maintain a history of the development process.
configuration	A collection of parameters, artifact references (such as datasets or models), and environment settings that are used to instantiate and run an execution or deployment of an executable or template. A configuration binds specific values to input parameters and specifies the versions of artifacts to be used. It is required to run an execution or deployment, and multiple configurations can be associated with a single executable or template (a 1:n relationship). Configurations are used in both training and serving processes to train models or serve predictions, respectively. They provide a way to customize and control the behavior of the executable or template for different scenarios or use cases.

Term	Definition
dataset	<p>A collection of data used for training, testing, or analysis in artificial intelligence and machine learning applications. A dataset has the following characteristics:</p> <ul style="list-style-type: none"> <li>• References a specific data source, such as a file, database, or API</li> <li>• Includes metadata describing the structure, format, and content of the data</li> <li>• May require specific tools, libraries, or credentials to access and process the data</li> <li>• Can be used as input to machine learning models, algorithms, or data processing pipelines</li> <li>• Is uniquely identified by an ID, which is used to reference the dataset in configurations and bind it to specific components or tasks</li> <li>• May be versioned to track changes and ensure reproducibility of results</li> <li>• Can be divided into subsets for training, validation, and testing purposes</li> <li>• May undergo preprocessing, cleaning, or augmentation steps to prepare the data for use in machine learning tasks</li> </ul>
deployment	<p>An instance of a model serving template (a serving executable or deployment template) that is configured to use a model artifact and apply it to data passed in a serving request (for example, for prediction). A successful deployment creates a model server and generates a deployment URL for inference. As input, a deployment takes one or more models and parameters from a configuration.</p> <p>The serving executable or deployment template defines the expected parameters and input dataset required for the serving process. Values for these parameters and input models are provided by a configuration.</p> <p>Deployments are implemented on a runtime that produces HTTPS endpoints, enabling secure access to the deployed models for inference.</p>
deployment template	<p>A template that defines the parameters and inputs required to serve or deploy one or more trained models. The template is instantiated with a specific configuration, which provides the necessary parameters and inputs. The resulting instance of the template is then used to serve the trained models. Deployment templates are used in the application to facilitate the deployment and serving of machine learning models.</p> <p>A deployment template is also known as a "serving executable".</p>
embedding	<p>A dense, low-dimensional vector representation that captures the semantic and syntactic information of a discrete input item, such as a word, phrase, or other entity, learned from a large dataset. Embeddings are typically used as input features in machine learning models to represent the underlying meaning and relationships of the items in a continuous space.</p>

Term	Definition
executable	<p>A reusable template that defines a workflow or pipeline for tasks such as training a machine learning model or creating a deployment. It contains placeholders for input artifacts (datasets or models) and parameters (custom key-pair values) that enable the template to be reused in different scenarios.</p>
	<p>There are two types of executables:</p>
	<ul style="list-style-type: none"> <li>• Non-deployable executables: When instantiated, they result in executions that may produce output artifacts.</li> <li>• Deployable executables: When instantiated, they result in deployments that generate URLs for inference.</li> </ul>
	<p>Executables are referred to as “templates” within the application. They can have user-defined labels applied to them, which are listed with the executable’s details.</p>
	<p>The code within an executable defines the training pipeline or model deployment pipeline, leveraging the placeholders for input artifacts and parameters to make the template reusable.</p>
execution	<p>A workflow execution, also known as a “run” in the app, is an instance of a non-deployable executable that represents a single run of a pipeline. There are two types of executions:</p>
	<ul style="list-style-type: none"> <li>• Training execution: A run of a training pipeline that takes input artifacts and parameters, and produces trained AI models as output artifacts.</li> <li>• Batch-inferencing execution: A run of a batch-inferencing pipeline that takes input artifacts and parameters, and produces result sets as output artifacts.</li> </ul>
	<p>Each execution is associated with metadata, including metrics, labels, tags, and custom information, which can be queried using the execution ID. For example, a training execution might have metrics like accuracy and loss, while a batch-inferencing execution might have metrics like processing time and number of processed records.</p>
function	<p>A modular, reusable piece of functionality that can be utilized across different scenarios or workflows. Functions are designed to perform specific tasks or computations and can be combined or integrated into larger scenarios to achieve complex business objectives. For example, a data preprocessing function can be used to clean and transform input data before feeding it into a machine learning model, while a model evaluation function can be used to assess the performance of a trained model. Functions promote code reusability, maintainability, and efficiency in building AI solutions.</p>
input artifact	<p>Placeholder in an executable or template that enables the attachment of datasets or models required for the execution of an AI workflow or pipeline. These artifacts can include training data, validation data, pre-trained models, or any other data or model assets needed for the AI process.</p>
job executable	<p>A simplified representation of a workflow executable. A job executable encapsulates the essential components and logic of a workflow, allowing users to easily understand and manage the execution of AI and machine learning tasks within the Functions Explorer interface of SAP AI Launchpad.</p>
job template	<p>A predefined configuration that specifies the parameters and resources required to execute a long-running AI process, such as model training, batch inference, or data preprocessing. It defines the Docker image, compute resources, environment variables, and other settings needed to run the job. Job templates act as blueprints that can be instantiated to create and execute specific job instances with customized parameters.</p>

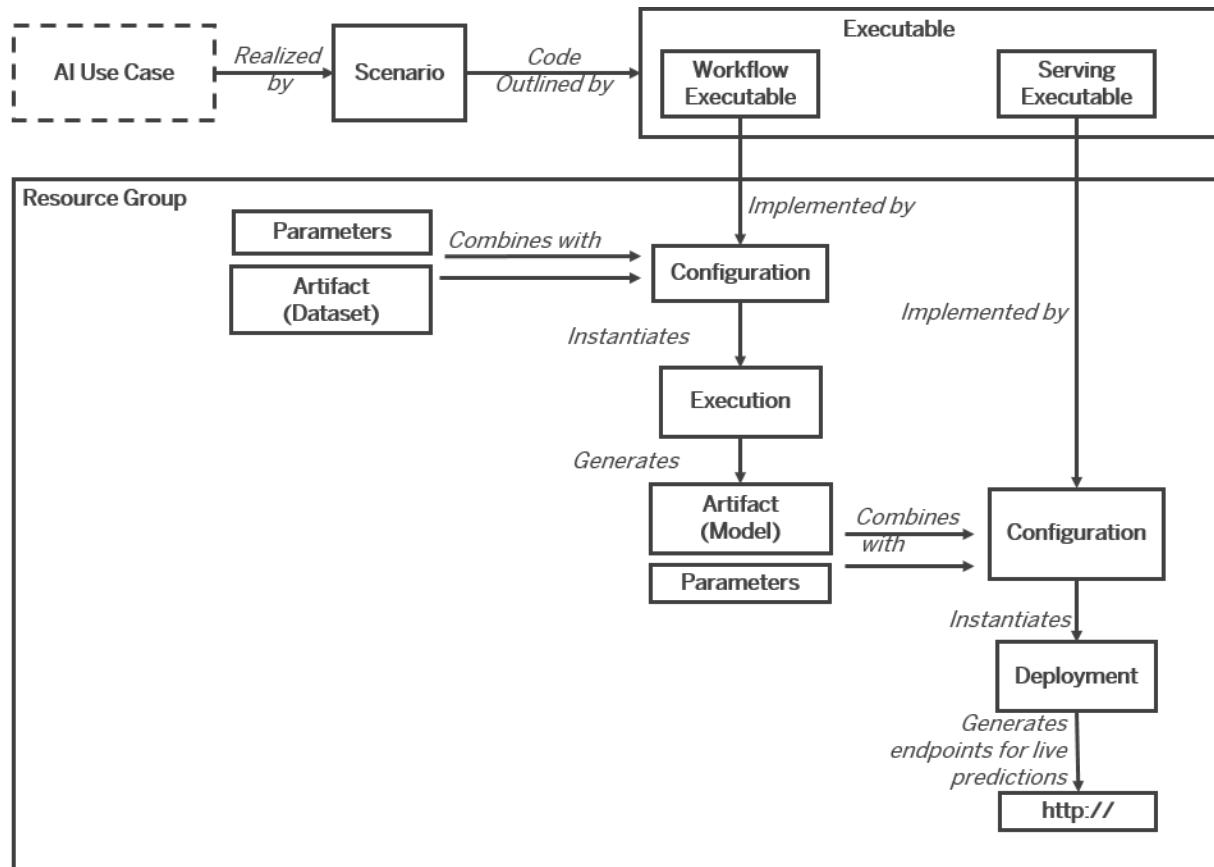
Term	Definition
knowledge graph	A structured representation of entities, their attributes, and the relationships between them, which can be inferred from various data sources, including structured, semi-structured, and unstructured data. Knowledge graphs enable machines to understand and reason about complex real-world concepts and their interconnections, facilitating tasks such as semantic search, recommendation systems, and question answering. Examples of knowledge graphs include ontologies, taxonomies, and knowledge bases.
label	A key-value pair attached to a metric to provide additional context and metadata. Labels are used to classify and categorize metrics, enabling users to filter, group, and aggregate data for better insights and analysis. A set of labels can be applied to each instance of a metric record, allowing for more granular and targeted monitoring.
metrics	A key/value pair where the value is numeric. Metrics are used to measure and monitor the performance, progress, and quality of a model during the training process. Common examples of metrics include accuracy, precision, recall, F1 score, and mean squared error. Metrics can have optional step, timestamp, and label fields, which provide additional information about the metric's context. Every metric (and associated labels), tag, and custom info must be associated with a specific training execution. This association allows for proper organization, tracking, and comparison of different training runs. Once the metric, tag, and custom info are saved, they can be queried using an execution ID. SAP AI Launchpad provides a user-friendly interface to visualize and analyze the captured metrics, enabling users to monitor the model's performance and make informed decisions based on the data.
model	<p>An artifact that is the output of a machine learning training process, representing the learned patterns, parameters, and structure of a trained system.</p> <ul style="list-style-type: none"> <li>A model is generated by a training process, which optimizes the model's parameters based on training data.</li> <li>A model consists of one or more files stored in a hyperscaler storage system (such as SAP AI Core's connected data storage) or the data lake for SAP HANA Cloud.</li> <li>Each model is uniquely identified by a model ID, which is used in a configuration to bind the model as an input artifact to the serving executable for deployment.</li> <li>Models can be manually uploaded to the connected data storage or automatically generated and uploaded as the output of a training execution in SAP AI Launchpad.</li> <li>The model encapsulates the learned knowledge of the AI system and is used for making predictions or decisions on new, unseen data during inference.</li> <li>Different types of models exist, such as neural networks, decision trees, or clustering models, each suited for different AI tasks and data types.</li> <li>In SAP AI Core, models are managed and versioned artifacts that can be deployed to serving endpoints for production inference.</li> </ul>
model serving template	A predefined configuration that specifies how a trained machine learning model is to be deployed and served for inference in a production environment, typically including details such as the model framework, runtime environment, resources required, and API endpoints.
operations	All activities within the AI lifecycle, including data preparation, model training, model deployment, application integration, model monitoring, and continuous improvement.

Term	Definition
output artifact	Generated results, typically AI models, produced by executing or running a training process or pipeline. When AI models are generated from an execution, they are automatically uploaded and registered to the connected data lake for SAP HANA Cloud or to the hyperscaler data storage, which is a scalable cloud storage solution provided by major cloud service providers such as Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP).
parameter	A placeholder in an executable (script, program, or model) to which a value of a specific data type (String, Integer, Float, Boolean, List, or Dictionary) is assigned during runtime. The values are provided by configurations, which are settings or files that define the parameter values. Parameters are used to provide input data, settings, or options to the executable for customizing its behavior in executions or deployments.
prompt	<p>A natural language instruction or query given to a generative AI model to elicit a response. An AI platform like SAP AI Launchpad includes capabilities for prompt experimentation, management, and administration:</p> <ul style="list-style-type: none"> <li>Prompt experimentation: Creating and running natural language prompts with a choice of large language models and adjustable parameters to test and optimize prompts.</li> <li>Prompt management: Saving, organizing, and managing prompts using collections, metadata (tags and notes), versioning to track changes over time, and deletion when no longer needed.</li> <li>Prompt administration: Tools for governing prompt usage, monitoring prompt performance, controlling access and permissions, and ensuring responsible AI practices.</li> </ul>
resource group	A dedicated workspace for a specific AI scenario or use case. It allows users to organize, manage, and collaborate on related AI entities such as configurations, executions, deployments, and artifacts within a defined scope. Resource Groups ensure isolation and access control, enabling users to work on their AI projects independently while sharing common resources and artifacts as needed. Each Resource Group is associated with a unique subscription to an SAP AI Launchpad tenant.
result set	A data artifact or dataset that contains the results of a batch inference run (execution). A batch inference run uses a trained machine learning model to process an input dataset containing data points or instances and generates predictions or inferences for each data point. The result set is an output artifact that stores these predictions or inferences.
run	A training process that generates a model or models based on a run template. Also known as an “execution”.
run template	A template that defines the components required for an AI pipeline within the ML Operations application. It specifies placeholders for parameters, input artifacts (such as datasets), and output artifacts (such as trained models) that are necessary to instantiate and run the executable successfully. The actual values for these placeholders are provided by a configuration, which is a set of settings and parameters specific to a particular use case. In SAP AI Launchpad, a run template is also referred to as a “workflow executable”.
runtime	The environment or platform that provides the necessary processing resources to execute AI and machine learning workloads, such as training and inference. It enables businesses to make their applications intelligent by leveraging AI and machine learning technologies, and allows them to train AI services using their data to automate tasks and processes. SAP AI Core is an example of a runtime.

Term	Definition
SAP AI Core	A service in the SAP Business Technology Platform designed to handle the execution and operations of your AI assets (machine learning models, datasets, and other AI-related resources) in a standardized, scalable, and hyperscaler-agnostic way, meaning it can work with different cloud service providers such as AWS, Azure, and Google Cloud Platform. SAP AI Core provides seamless integration with your SAP solutions, enabling you to easily incorporate AI capabilities into applications like SAP S/4HANA, SAP SuccessFactors, or SAP Customer Experience. Any AI function can be realized using popular open-source frameworks such as TensorFlow, PyTorch, or scikit-learn. SAP AI Core supports the full lifecycle management of AI scenarios, including data preparation, model training, deployment, monitoring, and retraining.
SAP AI Launchpad	A multitenant software as a service (SaaS) application on SAP Business Technology Platform (SAP BTP). Customers and partners can use SAP AI Launchpad as a centralized platform to manage AI use cases (scenarios), which are specific applications or problems that can be solved using AI, across multiple instances of AI runtimes (such as SAP AI Core). SAP AI Launchpad also provides access to the generative AI hub, which offers a set of pre-built generative AI models and tools that users can leverage to create and deploy their own generative AI applications.
scenario	<p>An implementation of a specific AI use case within a user's tenant. It consists of a pre-defined set of AI capabilities in the form of executables (pre-built AI models or services that can be directly deployed) and templates (outlines of AI models or services that need to be trained with customer data).</p> <p>A scenario can have multiple versions that correspond to different versions of its contained executables and templates. Placeholders within the executables and templates are populated with customer-specific values using a configuration.</p> <p>The purpose of a scenario is to group executables and templates related to an AI use case and make them available to all AI consumers in the tenant, while also enabling version tracking as the AI models and services evolve over time.</p>
serving executable	<p>A predefined AI pipeline that encapsulates the necessary components and logic to deploy an AI model or application. It serves as a blueprint for creating a specific deployment instance.</p> <p>A deployment template defines placeholders for parameters, input artifacts (such as datasets), and output artifacts (such as trained models). These placeholders are filled with actual values provided by a configuration when the template is instantiated.</p> <p>Instantiating a deployment template involves providing the required configuration, which specifies the concrete values for the parameters and input artifacts. This process creates a deployment instance that can be executed to train, evaluate, or serve an AI model or application.</p> <p>In the SAP AI Launchpad, deployment templates are referred to as "serving executables".</p>
tag	A name/value pair that is used to categorize and organize test executions. Tags allow for the segregation and grouping of test executions based on specific criteria, making it easier to manage, analyze, and report on the results. For example, you can assign tags to a group of selected test executions to indicate their purpose, priority, or other relevant characteristics. A set of tags can be associated with a MetricResource, which is an entity that represents a specific metric or measurement. The MetricResource, in turn, is linked to an execution, establishing a connection between the tags and the corresponding test run. This relationship enables the effective tracking, monitoring, and evaluation of test executions based on their assigned tags.

Term	Definition
tenant	A logically separated customer instance that represents an organization or a company. Each tenant has its own isolated collection of customized content and services, which are available only to that specific tenant, its users, and the service provider managing the multi-tenant environment.
training	The process of running a machine learning algorithm on a dataset to produce a trained model. The trained model is an artifact that captures the patterns and relationships learned from the data, which can then be used to make predictions or decisions on new, unseen data.
workflow executable	A template that defines the components required for an AI pipeline within the ML Operations application. It specifies placeholders for parameters, input artifacts (such as datasets), and output artifacts (such as trained models) that are necessary to instantiate and run the executable successfully. The actual values for these placeholders are provided by a configuration, which is a set of settings and parameters specific to a particular use case. In SAP AI Launchpad, a workflow executable is also referred to as a "run template".

You can see how these concepts interact from the following diagram:



## 3.1 SAP AI Core Overview

SAP AI Core is the key to integrating artificial intelligence capabilities in your SAP solutions.

Specifically, SAP AI Core helps you to:

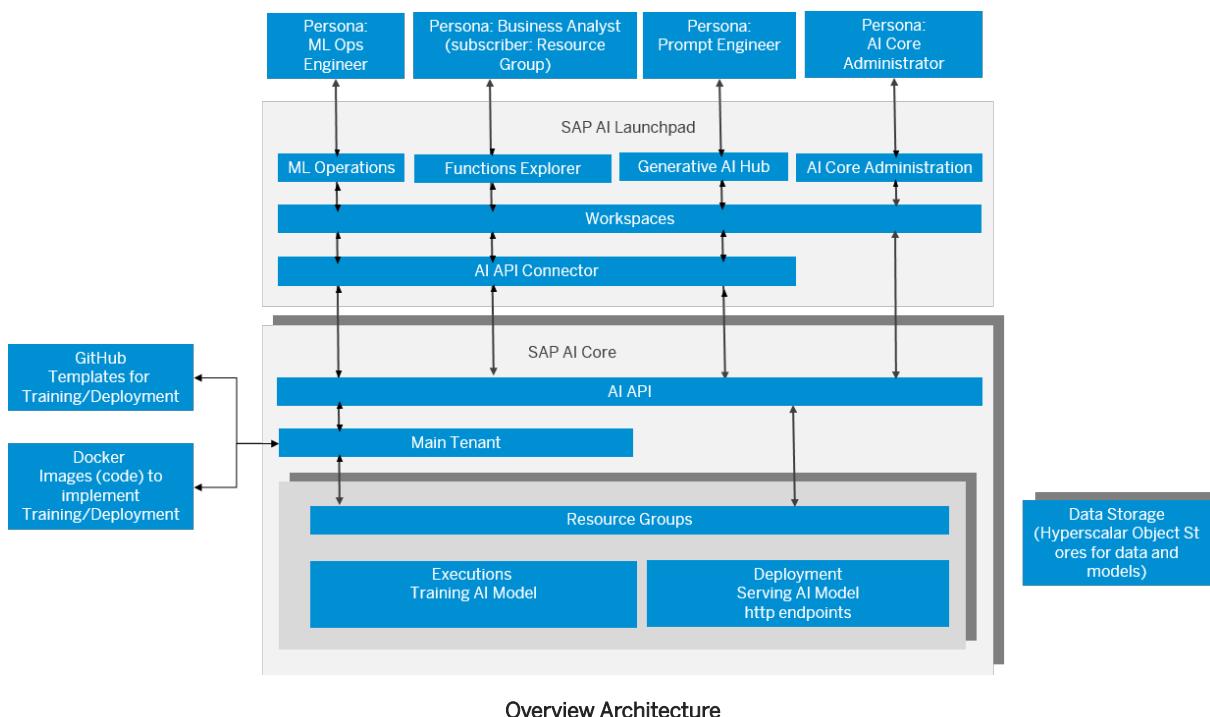
- Seamlessly and easily embed AI capabilities into other applications
- Leverage high-volumes of data from applications to create robust AI learning models
- Execute AI training on accelerated hardware
- Serve AI inference with low latency and high-throughput in a cost-efficient manner
- Adhere to a compliant, explainable, and maintainable process
- Manage all stages of the AI lifecycle using a comprehensive set of tools and services
- Focus on the productization and operationalization of AI scenarios

Management and operations of your AI content (such as versioning, deployment, and monitoring) is unified across your SAP solutions. Authoring of your AI content is, however, open to different toolsets (such as JupyterLab).

The SAP AI Core service is intended to be used together with SAP AI Launchpad and the **AI API**. The key components are SAP AI Core, SAP AI Launchpad, and the **AI API**.

- **SAP AI Core** provides an engine that lets you run AI workflows and model serving workloads.
- **SAP AI Launchpad** manages a number of AI runtimes. It allows various user groups to access and manage their AI scenarios.
- **AI API** provides a standard way of managing the AI scenario lifecycle on different runtimes, regardless of whether they are provided on SAP technology (such as SAP S/4HANA) or on partner technology (such as Amazon Web Services). When the AI API is deployed on runtimes other than SAP AI Core, the runtimes have to provide a runtime adapter.

[Overview Architecture \[page 47\]](#) shows these three main components in an overview architecture diagram.



Overview Architecture

### 3.1.1 SAP AI Core Systems Overview

Your SAP AI Core system connects internal and external tools.

Users interact with the various repositories, systems, and objects when working with SAP AI Core; some of these objects are provided by SAP. In other cases, customers provide these components to enable enhanced control (authorizations) and continuous integration/continuous deployment (CI/CD).

### Key Repositories, Systems, and Objects

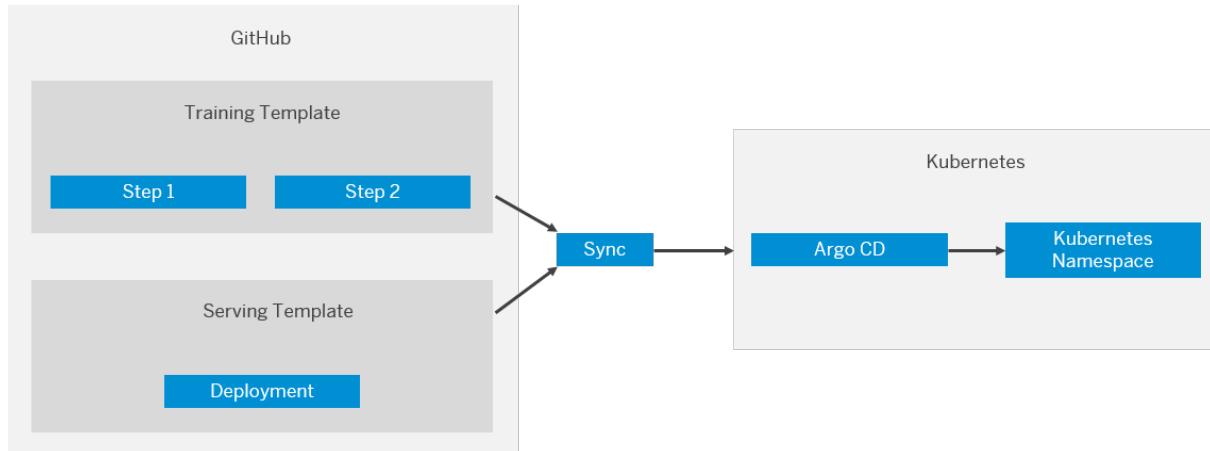
Summary of Elements and Their Use

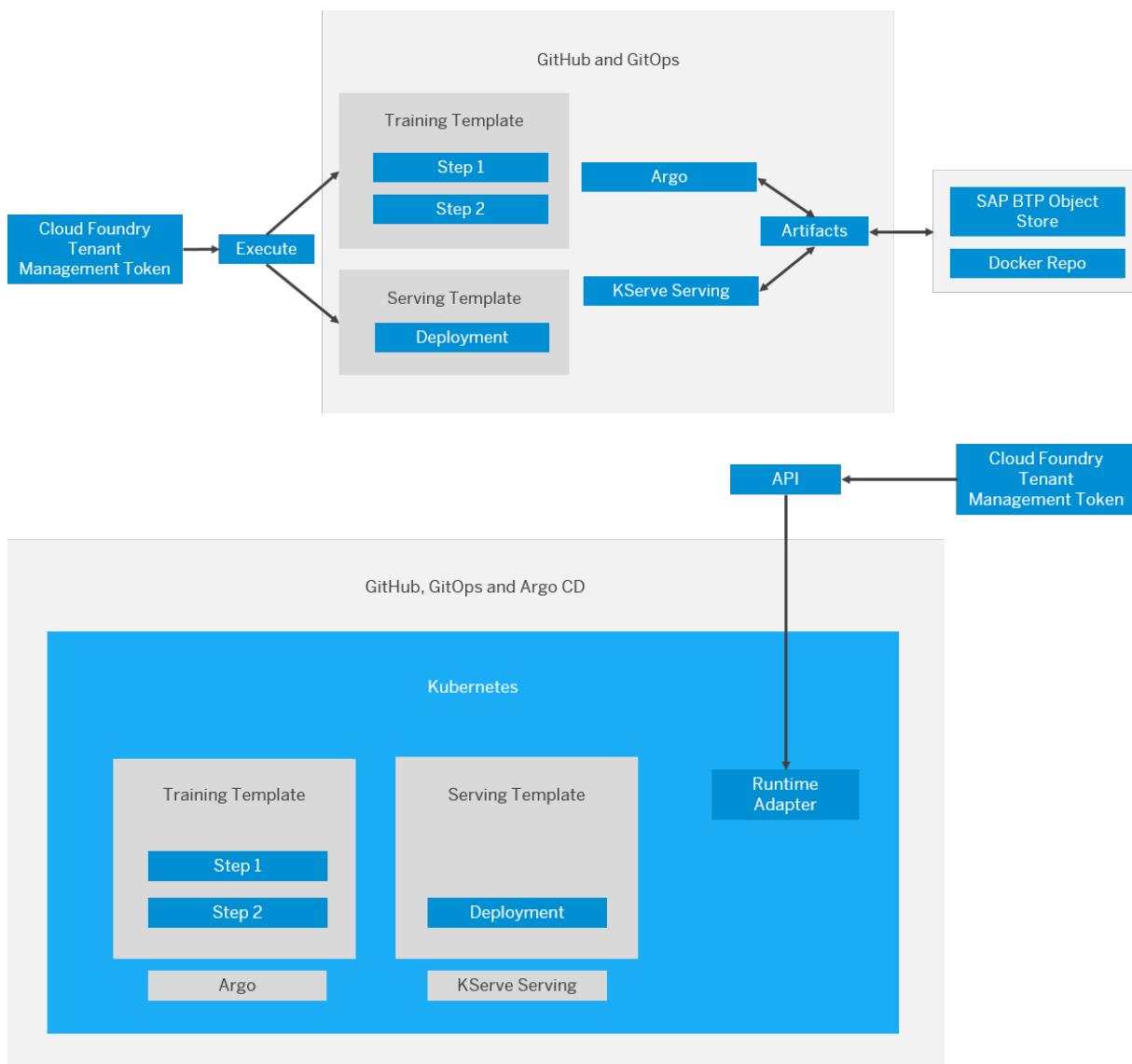
What	Why
Git repo	For storing training and serving workflows and templates
Hyperscaler storage	For storage of input and output artifacts, such as training data and models (for example, SAP BTP Object Store Service)
Docker repo	For custom Docker images referenced in the templates
Kubernetes (K8s)	The K8s cluster orchestrates and scales the pods, which are used in AI pipelines. Resource group isolation is based on a K8s namespace.
KServing (K)	For optimized deployments of machine learning models. Deployment templates use KServe notation.

What	Why
AI API	For managing your artifacts and workflows (such as training scripts, data, models, and model servers) across multiple runtimes
	<p><b>① Note</b></p> <p>The AI API can also be used to integrate other machine learning platforms, engines or runtimes into the AI ecosystem.</p>
Argo Workflows	A container native, workflow engine for Kubernetes.
SAP AI Launchpad	SAP AI Launchpad is a multitenant software as a service (SaaS) application on SAP Business Technology Platform (SAP BTP). Customers and partners can use SAP AI Launchpad to manage AI use cases (scenarios) across multiple instances of AI runtimes (such as SAP AI Core). SAP AI Launchpad also provides generative AI capabilities via the Generative AI Hub.

## Synchronization

- Your GitOps implementation is integrated with SAP AI Core to enable CI/CD.
- Templates are synchronized from your Git repository to a Kubernetes cluster.
- Your training and serving templates are regularly synchronized (every few minutes).
- SAP AI Core checks the template syntax, and if synchronization fails, then error messages are displayed.

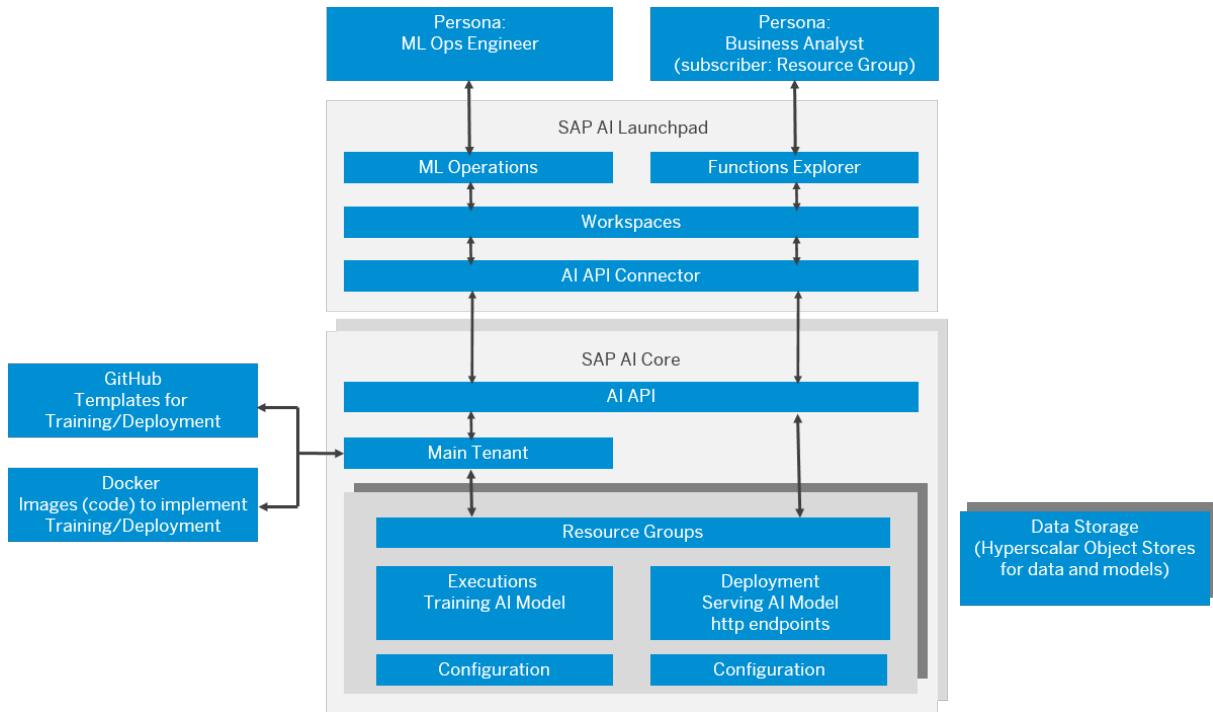




## Process Summary

- An SAP BTP token is used in **AI API** for authentication of the API calls.
- Users run templates via the **AI API**.
- Training templates are executed using the **Argo** workflow. The training pipeline consumes data input artifacts and outputs a model artifact.
- An execution is created using the training template and **AI API** configuration. The result is a training job.
- Serving templates are deployed using **KService**.
- A deployment is created using the serving template and **AI API** configuration. The result is an inference server.
- Artifacts (such as datasets, models) are copied to and from the hyperscaler storage.
- Required images are pulled from the registered Docker repository.

- Details about scenarios and executables are retrieved from the Kubernetes cluster by the AI API.



### 3.1.2 AI API Overview

The AI API lets you manage your AI assets (such as training scripts, data, models, and model servers) across multiple runtimes.

Argo workflows and serving templates, as well as their execution and deployment, are managed using the SAP AI Core implementation of the AI API. In SAP AI Core, the Argo workflow and serving templates are mapped under the concept of `Executable`. For the mapping mechanism to work, the Argo workflows and serving templates require certain attributes in the metadata section of the `YAML` file. These attributes are shared by both template types.

SAP AI Core provides additional APIs that are runtime-specific. These are available in the AI Core API specification, which is an extension of the AI API specification.

### Related Information

[AI Core API](#)

[AI API](#)

# AI API Runtime Implementations

The AI API specification is a general specification for the lifecycle management of machine learning artifacts. SAP AI Core is one specific runtime implementation of the AI API specification. It is also possible to provide other runtime implementations of the AI API specification, independent of SAP AI Core. This section describes the necessary boundary conditions and implementation requirements.

The benefit of using AI API is that clients can integrate with all AI API-enabled runtime implementations. For example, SAP AI Launchpad can interact with custom runtime implementations as long as the same APIs are provided. Intelligent Scenario Lifecycle Management can also integrate with AI API-enabled runtimes. The SAP AI SDK Base (Python) can also be used (for more information, see [sap-ai-sdk-core](#)).

## AI API Specification

The AI API specification comprises the following parts:

- A main specification
- Extensions:
  - Analytics extension
  - Resource group extension
  - Dataset management extension
  - Metrics extension

### → Recommendation

Implement at least the main specification, and then implement the extension specifications based on your use case.

## AI API Runtime Capabilities Endpoint

`Meta API` is part of the AI API specification (endpoint `/lm/meta`). The implementation must return a configuration response that specifies the capabilities of the AI API runtime implementation.

`Meta API` allows AI API clients to query the capabilities of an AI API implementation so that they can select which commands or user interfaces are available. For example, some AI API runtimes may offer executions but not deployments. They may also offer logs for executions and not for deployments. As an example, if a client of SAP AI Core such as SAP AI Launchpad queries the `Meta API` endpoint of SAP AI Core, the response will be for example:

```
json
{
  "aiApi": {
    "capabilities": {
      "logs": {
        "deployments": true,
        "executions": true
      }
    }
  }
}
```

```

        },
        "multitenant": true,
        "shareable": true,
        "staticDeployments": true,
        "timeToLiveDeployments": true,
        "userDeployments": true,
        "userExecutions": true
        "executionSchedules": true
    },
    "limits": {
        "deployments": {
            "maxRunningCount": -1
        },
        "executions": {
            "maxRunningCount": -1
        },
        "minimumFrequencyHour": 1,
        "timeToLiveDeployments": {
            "minimum": "10m",
            "maximum": -1
        }
    },
    "version": "2.18.0"
},
"extensions": {
    "analytics": {
        "version": "1.0.0"
    },
    "metrics": {
        "capabilities": {
            "extendedResults": true
        },
        "version": "1.0.0"
    },
    "resourceGroups": {
        "version": "1.2.0"
    }
},
"runtimeApiVersion": "2.21.0",
"runtimeIdentifier": "aicore"
}

```

SAP AI Launchpad and other clients can then react accordingly and hide the deployments on the user interface for this runtime implementation of AI API.

Capabilities include the following:

Capability	When true, allows users to:
logs.executions	View logs for an execution
logs.deployments	View logs for a deployment
multitenant	Use SAP AI Launchpad as a main tenant user (supports resource groups)
shareable	Clients can share one instance
staticDeployments	Static, always running endpoints for inference are available without the user having to start a deployment
userDeployments	Stop, update, or delete a deployment
userExecutions	Stop or delete an execution

Capability	When true, allows users to:
timeToLiveDeployments	The runtime engine allows defining the time until a deployment is automatically deleted
analytics	Review summary information for all tenants
bulkUpdates	Stop or delete up to 100 executions or deployments at once
executionSchedules	Create schedules

Limits include the following:

Limit	Details
deployments.maxRunningCount	Limits the number of running concurrent deployments in a resource group, if any
executions.maxRunningCount	Limits the number of running concurrent executions in a resource group, if any
timeToLiveDeployments.minimum	The minimum possible value for the ttl parameter in a deployment, if supported
timeToLiveDeployments.maximum	The maximum possible value for the ttl parameter in a deployment, if supported
minimumFrequencyHour	The minimum possible value for schedule of an execution, if supported

In addition to the general AI API specification, there is also a number of extensions that cover additional use cases. These might not be implemented in all runtime engines.

The extensions are:

Extension	Details
analytics	The analytics extension contains endpoints for fetching analytical information of a resource group or tenant
metrics	The metrics extension contains endpoints for writing to and reading from metrics endpoints, to store and retrieve metrics generated during executions
resourceGroups	The resource group extension contains endpoints for managing resource groups
dataset	The dataset extension contains endpoints for uploading and downloading files

## Related Information

[Serving Templates \[page 513\]](#)

[Workflow Templates \[page 464\]](#)

[AI API Specification](#)

[Custom Runtime Capabilities Using the Meta API](#)

[Analytics Extension](#)

[Resource Groups Extension](#)

[Intelligent Scenario Lifecycle Management](#)

### 3.1.3 Resource Groups

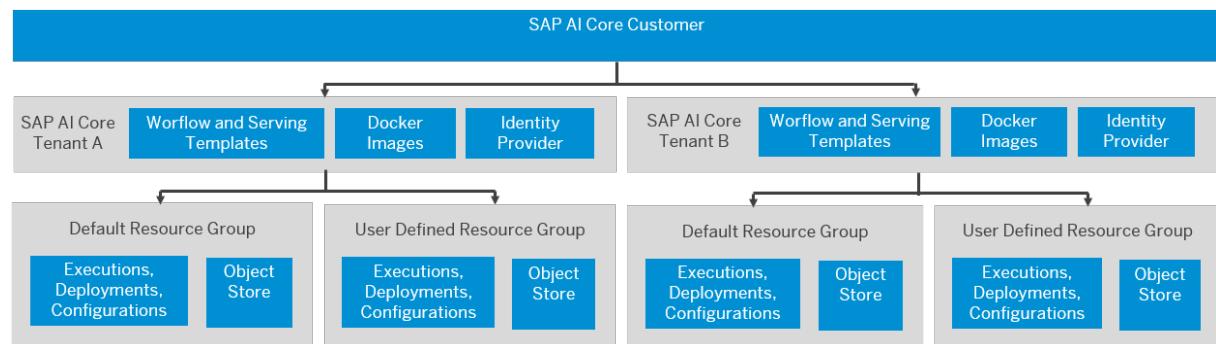
SAP AI Core tenants use resource groups to isolate related ML resources and workloads. Scenarios, executables, and Docker registry secrets are shared across all resource groups.

Resource groups represent a virtual collection of related resources within the scope of one SAP AI Core tenant. When your tenant is onboarded, a default resource group is created immediately. Further resource groups can be created, or deleted by your tenant administrator with the AI API. Tenants can map the resource groups based on the corresponding usage scenarios.

If your SAP AI Core tenant uses resource groups to isolate the scenario consumer tenant and the resource groups are subsequently deleted, the scenario consumers will be deprovisioned. SAP AI Core is not aware of the scenario consumer of the tenant. The standard XUSAA multitenancy model is followed.

#### 3.1.3.1 Scope of Resources

Resources that are available for tenants and resource groups differ based on the available scope.



#### Tenant-Level Resources

Tenant-level resources include:

- Workflow templates
- Serving templates
- Docker registry (containing the Docker images)
- User authentication and authorization (UAA)

User authentication and authorization is based on the SAP AI Core tenant. The tenant is the holder of the access token obtained using the SAP AI Core service key. The SAP AI Core tenant can set the resource group

in the request header at runtime, or during lifecycle management, using the AI API. If the resource group is not set, the default resource group is used.

## Resource Group-Level Resources

Executables at tenant level are shared across all of the resource groups. At resource group level, the object store is registered by setting the resource group header.

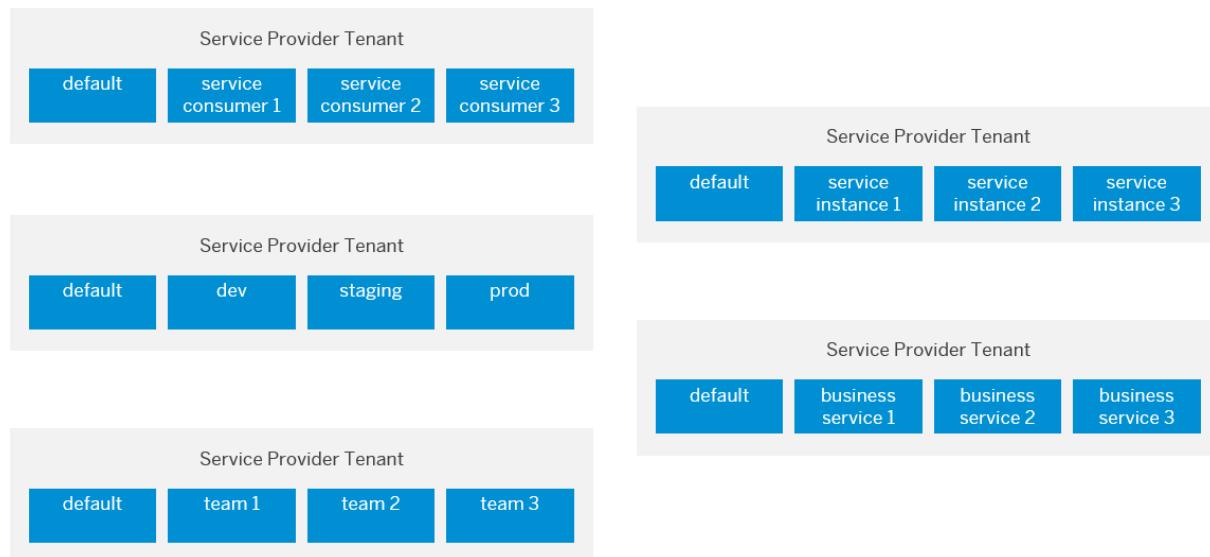
SAP AI Core tenants must consider security aspects in the design of AI functions.

### → Recommendation

Do not use the same object store bucket with the same AWS IAM user for multiple resource groups.

Runtime entities such as executions, deployments, configurations, and artifacts belong to specific resource groups and cannot be shared across resource groups.

## Examples of Resource Group Mapping



## 3.2 Generative AI Hub in SAP AI Core Overview

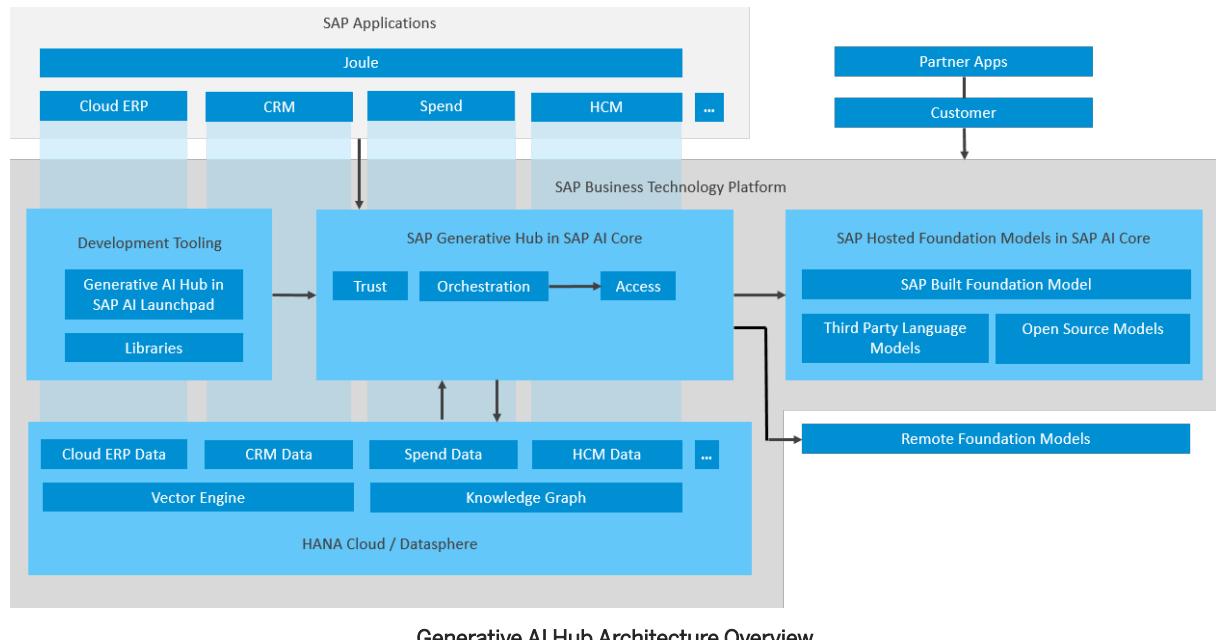
The generative AI hub incorporates generative AI into your AI activities in SAP AI Core and SAP AI Launchpad.

Generative AI models are self-supervised, deep learning models trained on vast amounts of unlabeled data. They use AI technology and industrial-scale computational resources to learn complex patterns and semantic knowledge bases. These models excel in tasks like natural language processing (NLP). By parsing inputs, such

as prompts, and predicting target words, they return contextually relevant responses in natural language. A single model can handle multiple tasks by using different input formats and output modes.

Generative AI models are general by design, but you can fine-tune them with additional embeddings. In this way, you can make them suitable for specialized or domain-specific use cases.

SAP AI Core and the generative AI hub help you to integrate LLMs and AI into new business processes in a cost-efficient manner.



# 4 Initial Setup

Get started with SAP AI Core using the standard procedures for the SAP BTP, Cloud Foundry environment or Kyma environment.

## Related Information

[Enabling the Service in Cloud Foundry \[page 57\]](#)

[Enabling the Service in the Kyma Environment \[page 78\]](#)

## 4.1 Enabling the Service in Cloud Foundry

Enable SAP AI Core using the standard procedures for the SAP BTP, Cloud Foundry environment.

### → Tip

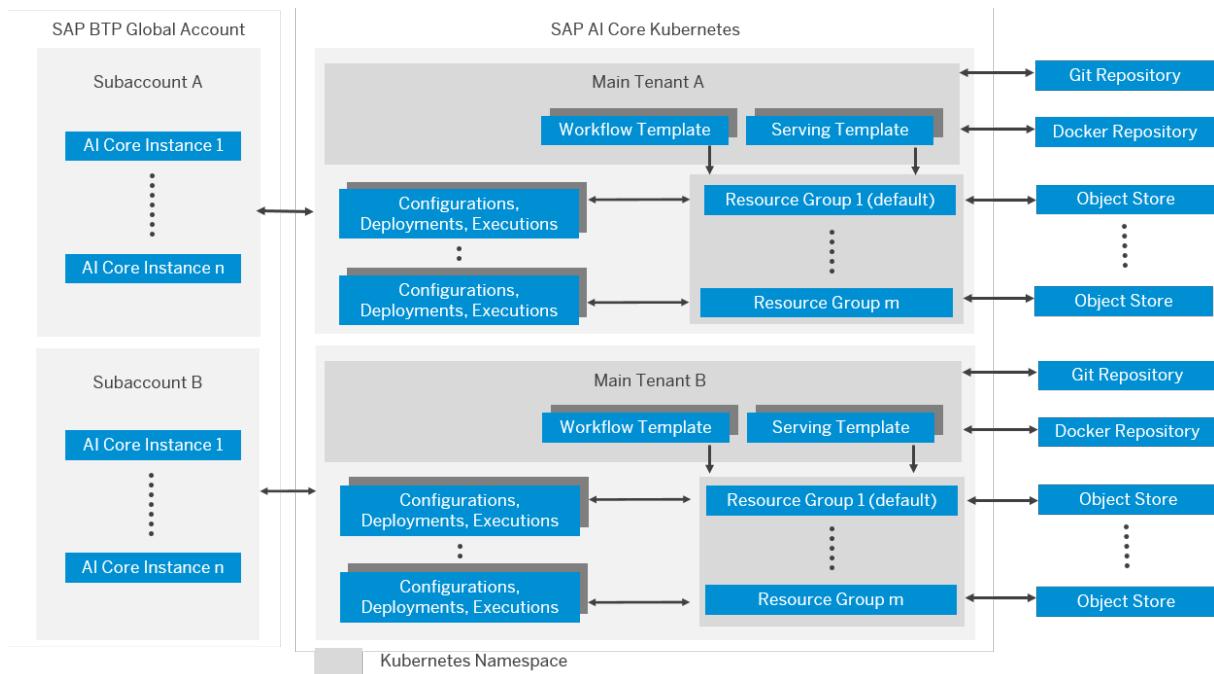
You can use the booster [Set Up Account for SAP AI Core](#) to automate the steps described in this section on the SAP BTP cockpit. For more information, see [Use Boosters for Free Tier Use of SAP AI Core and SAP AI Launchpad](#).

When you provision SAP AI Core from the SAP BTP cockpit in SAP Business Technology Platform, the system generates a service key. This key contains the URLs and credentials you need to access the SAP AI Core instance.

SAP AI Core is a tenant-aware reuse service that isolates tenants based on the zone ID, which represents the subaccount. When you create an SAP AI Core service instance within a subaccount, it represents an SAP AI Core tenant.

### ⓘ Note

The SAP AI Core service doesn't isolate tenants based on the service instance ID. If you create multiple service instances within the same subaccount, they all reference the same SAP AI Core tenant.

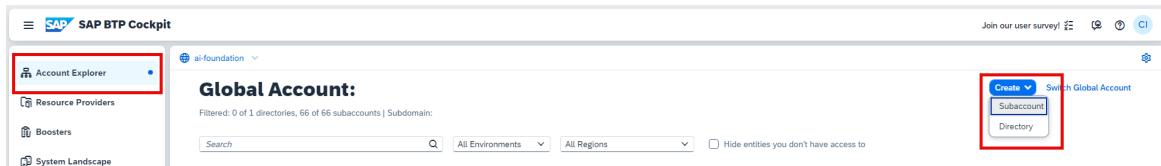


## 4.1.1 Create a Subaccount

Create a subsaccount in your global account using the SAP BTP cockpit.

### Procedure

- In the SAP BTP cockpit, choose *Account Explorer* and then click *Create Subaccount*.



- In the *Create Subaccount* dialog, enter a name for your subaccount and select the region.

The parent defaults to the name of your global account.

### Create Subaccount

Display Name\*  
My Subaccount

Description  
Enter a description of up to 300 characters

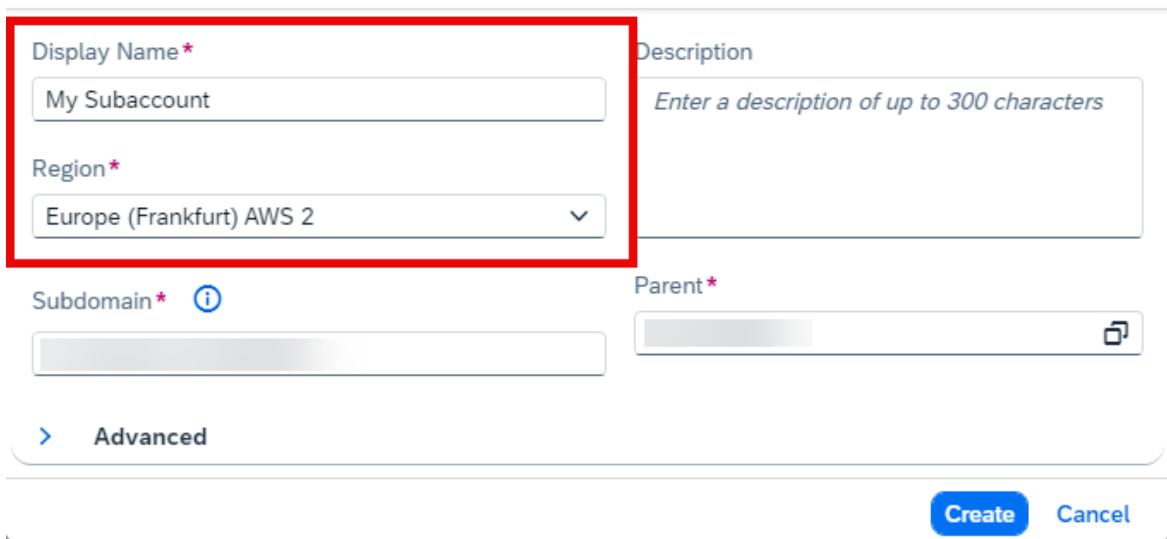
Region\*  
Europe (Frankfurt) AWS 2

Subdomain\* ⓘ

Parent\* ⓘ

> Advanced

**Create** **Cancel**



3. **Optional:** If your subaccount is used for production purposes, under *Advanced* select the *Used for production* checkbox.
- This setting does not change the configuration of your subaccount. It is intended to help you manage the production subaccounts in your global account. For example, your cloud operator can refer to it when handling incidents related to mission-critical accounts.

< Advanced

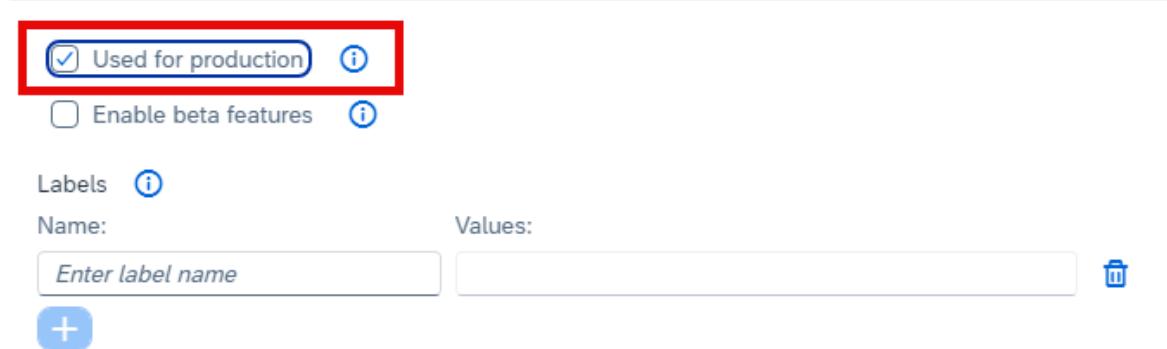
Used for production ⓘ

Enable beta features ⓘ

Labels ⓘ

Name:  Values:  ⌂

+



4. Click *Create*.
5. Return to the *Account Explorer* to view your subaccount.

SAP BTP Cockpit

ai-foundation

## Global Account

Filtered: 0 of 1 directories, 67 of 67 subaccounts

Search

Directories and Subaccounts Subaccounts (67)

Name

My Subaccount

## Related Information

[SAP BTP Cockpit](#)

### 4.1.2 Enable Cloud Foundry

#### Procedure

1. Click your subaccount and on the [Overview](#) page, choose [Enable Cloud Foundry](#).

The screenshot shows the SAP BTP Cockpit interface. On the left, a sidebar menu includes 'Overview' (highlighted with a red box), 'Services', 'Service Marketplace', 'Instances and Subscriptions', 'HTML5 Applications', 'Connectivity', 'Security', 'Entitlements' (highlighted with a red box), and 'Usage Analytics'. The main content area is titled 'Subaccount: My Subaccount - Overview' under 'My Subaccount'. It displays the following details:

- General:** Entitlements (30), Instances and Subscriptions (0).
- Cloud Foundry Environment:** Subdomain: [redacted], Tenant ID: [redacted], Subaccount ID: [redacted]. Provider: Amazon Web Services (AWS), Region: Europe (Frankfurt) AWS 2, Environment: Multi-Environment.
- Cloud Foundry Environment Status:** You are currently not using Cloud Foundry capabilities. A red box highlights the 'Enable Cloud Foundry' button.

- Enter the basic information for your Cloud Foundry environment instance and click *Create*.

### 4.1.3 Create a Space

#### Procedure

- On the overview page for your subaccount, choose *Create Space*.

The screenshot shows the 'Subaccount: Testsubaccount - Overview' page. At the top right is an 'Edit' button. Below it, the subaccount details are listed:

- General: Entitlements (13), Instances & Subscriptions (1).
- Cloud Foundry Environment: Subdomain: [redacted], Tenant ID: [redacted], Subaccount ID: [redacted]. Provider: Amazon Web Services (AWS), Region: Europe (Frankfurt), Used for Production: Yes, Beta Features: Disabled, Created On: Aug 25, 2021, 22:14, Modified On: Aug 25, 2021, 22:15.
- Cloud Foundry Environment (Details): Org Name: [redacted], API Endpoint: [redacted], Org ID: [redacted]. Manage environment instance, Disable Cloud Foundry.
- Cloud Foundry Environment (Spaces): Spaces (0). A red box highlights the 'Create Space' button.

- Enter a name for your space, assign the required roles, and click *Create*.

**Create Space**

Space Name:\*

Test

Assign space roles to:

- Space Manager
- Space Developer
- Space Auditor

**Create** **Cancel**

## Related Information

[About Roles in the Cloud Foundry Environment](#)

### 4.1.4 Add a Service Plan

Configure the required entitlements to make SAP AI Core accessible in your subaccount.

## Procedure

- In SAP BTP cockpit, navigate to your global account and choose *Entitlements* and then *Entity Assignments*.
- Use the input help to select your subaccount and click *Edit*.
- Choose *Edit*.

**Global Account**

Subaccounts/Directories:

Search  My Subaccount

Service	Service Technical Name	Plan	Set Quota Limit	Quota Assignment	Remaining Quota	Actions
[DEPRECATED] Audit Log Retrieval	auditlog-api	default	<input type="button"/>	Assigned (1 unit)	<input type="button"/>	<input type="button"/>
Autoscaler	autoscaler	standard	<input type="button"/>	Assigned (1 unit)	<input type="button"/>	<input type="button"/>

4. Choose [Add Service Plans](#).

The screenshot shows the SAP BTP Cockpit interface. In the top navigation bar, under 'Services', it says 'Subaccount: Testsubaccount - Entitlements'. Below the navigation bar, there's a search bar and a red box highlights the 'Add Service Plans' button.

5. Select SAP AI Core and pick your chosen service plan or plans.

This screenshot shows the 'Add Service Plans' dialog for SAP AI Core. On the left, under 'Services available to this subaccount', 'SAP AI Core' is selected and highlighted with a red box. On the right, under 'Available Plans', three options are listed: 'extended', 'standard', and 'free'. The 'extended' plan is checked and described as the default plan for aicore and gen ai features. The 'standard' plan is also checked and described as the default standard plan. The 'free' plan is checked and described as a free plan with no usage limits, noting that only community support is available for free service plans. A red box highlights the 'extended' plan. At the bottom right, a red box highlights the 'Add 2 Service Plans' button.

#### → Tip

To use generative AI capabilities, choose the extended plan. For more information, see [Service Plans \[page 64\]](#).

6. Save your changes.

This screenshot shows the 'Testsubaccount (Unsaved changes)' table. It lists various services and their assigned plans. The 'Audit Log Viewer' has a 'free (Application)' plan assigned. 'Cloud Identity Services' has an 'application' plan assigned. 'Cloud Integration Automation Service' has a 'standard (Application)' plan assigned. 'Content Agent Service' has an 'application' plan assigned. 'HTML5 Application Repository Service' has an 'app-host' plan assigned. 'Master Data Integration (Orchestration)' has a 'standard (Application)' plan assigned. A red box highlights the 'Save' button at the top right of the table.

## Related Information

[Service Plans \[page 64\]](#)

## 4.1.4.1 Service Plans

The SAP AI Core service plan you choose determines pricing, conditions of use, resources, available services, and hosts.

Your choice depends on your use case:

- Use the *Free* plan to explore the service with limited resources and community-only support.
- Use the *Standard* plan for production workloads without generative AI.
- Use the *Extended* plan for production workloads with generative AI hub access.

## Comparison of Service Plans

Feature / Plan	Free (Exploration)	Standard (Production)	Extended (Production + Generative AI)
Account type	Enterprise account (SAP BTP free tier only; not available in SAP BTP trial)	Enterprise account	Enterprise account
Support	Community support only (no SLA)	Full SAP support with SLA	Full SAP support with SLA
Generative AI hub	✗ Not included	✗ Not included	✓ Included
Pricing	Free	Pay per resource + baseline charge	Pay per resource + model/token usage
Instances per subaccount	1 instance (mutually exclusive with Standard)	1+ instances (mutually exclusive with Free)	1+ instances
Executions/Deployments	1 running execution or deployment at a time	Multiple	Multiple
Resource groups	Default group only	Multiple (quota applies)	Multiple (quota applies)
Resource plan	Starter plan only	Choice of resource plans	Choice of resource plans
Upgrade/downgrade rules	Upgrade from Free to Standard is possible.  Downgrade from Standard to Free is not possible.	Cannot be created if a Free plan is active in the same subaccount.	Cannot be created if a Free plan is active in the same subaccount.

### ⓘ Note

You can only run either a Free or a Standard plan in the same subaccount — not both. Extended follows the same rules as Standard.

For supported regions, see [SAP Discovery Center](#).

## Quotas

### Deployment Quotas

Each tenant is assigned a default quota that limits the number of deployments and replicas per deployment. If you reach this quota, your deployment will not be created and you will be notified accordingly. You can free up your quota by deleting existing deployments.

Alternatively, you can request an increase to your quota by creating a ticket on component **CA-ML-AIC**. Enter the description **Request to Increase Quota** and include details about the size of your increase, whether you want to include deployments, replicas, or both, and your subaccount ID.

### Resource Group Quotas

#### ⚠ Restriction

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 \[page 95\]](#).

### Tenant-Wide Generic Secrets Quotas

Each tenant can have a maximum of five tenant-wide secrets. If you reach this limit, you receive an error message. To free up space, delete tenant-wide secrets as described at [Delete a Generic Secret \[page 122\]](#). Alternatively, submit a ticket to request an increase in your quota.

## Related Information

[Set Up the Free Plan \[page 66\]](#)

[SAP Discovery Center](#)

[SAP BTP Service Description Guide](#)

[Choose an Instance \[page 462\]](#)

## 4.1.4.1.1 Set Up the Free Plan

The free plan lets you try out SAP AI Core for testing and familiarization purposes at no cost.

### Prerequisites

You have a global account in the free tier model for SAP BTP (not available in SAP BTP Trial).

### Context

Alternatively, you can use a booster to set up the free plan for SAP AI Core. For more information, see the tutorial [Use Boosters for Free Tier Use of SAP AI Core and SAP AI Launchpad](#).

### Procedure

1. Open your global account in the SAP BTP cockpit.
2. Go to your subaccount.
3. In the navigation area, choose *Instances and Subscriptions*.
4. Choose *New Instance or Subscription*.
5. In the *Service* field, search for SAP AI Core.
6. In the *Plan* field, choose *Free*.

The screenshot shows the 'New Instance or Subscription' wizard with three steps: 1. Basic Info, 2. Parameters, and 3. Review. Step 1 is active. It has fields for Service (SAP AI Core) and Plan (standard or free). A red arrow points to the 'free' option in the dropdown menu, which is highlighted with a blue border.

## Results

A Free plan tenant is created in your subaccount.

## Related Information

[Service Plans \[page 64\]](#)

[Using Free Service Plans](#)

[Getting a Global Account](#)

### 4.1.4.1.2 Upgrade a Service Plan

Upgrade your SAP AI Core service instance from the free plan to a standard or extended plan while keeping your data and models.

## Context

During the upgrade, all metadata and transaction data, including trained models, is retained.

You can also upgrade from the standard plan to the extended plan.

#### ⚠ Restriction

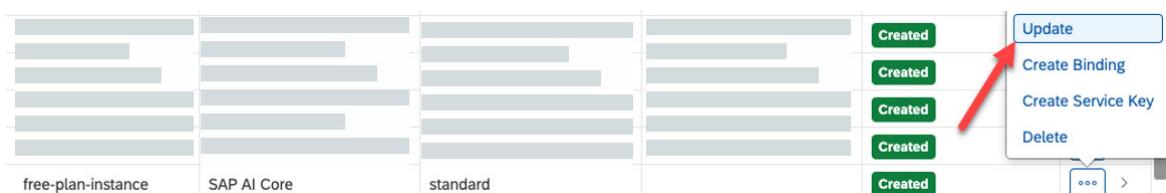
You cannot downgrade from the standard or extended plan to the free plan.

Downgrading from the extended plan to the standard plan is also not supported.

If a standard or extended instance is deleted, you cannot create a new standard plan instance.

## Procedure

1. Open your global account in the SAP BTP cockpit.
2. Go to your subaccount.
3. In the navigation area, choose *Instances and Subscriptions*.
4. Search for SAP AI Core.
5. At the end of the subscription row, select the ellipsis (...) and choose *Update*.



- In the wizard that opens, select *default* and click *Update Subscription*.

## Results

Free plan restrictions no longer apply.

All data from your free plan is migrated automatically to your new plan.

User permissions remain unchanged.

## 4.1.5 Create a Service Instance

### Procedure

- In SAP BTP cockpit, navigate to your subaccount within your global account and choose *Service Marketplace*.

You will see a tile for SAP AI Core.

The screenshot shows the SAP BTP Cockpit interface. The left sidebar has 'Services' expanded, with 'Service Marketplace' selected. The main area displays a grid of service tiles under 'Subaccount: Testsubaccount - Service Marketplace'. One tile, 'SAP AI Core', is highlighted with a red box. The tile description reads: 'AI Core provides a modern runtime to execute machine learning workloads such as training...'. Below the tiles, it says 'Extension Suite - Digital Experience'.

- Open the tile and click *Create*.

The screenshot shows the SAP BTP Cockpit interface for creating a SAP AI Core instance. The left sidebar has 'Services' expanded, with 'Service Marketplace' selected. The main area shows the 'SAP AI Core' tile from the previous screen. A red box highlights the 'Create' button in the top right corner of the service card. The service card details the SAP AI Core service, including its name (aicore) and a brief description: 'AI Core provides an infrastructure for a number of machine learning applications and services, including the option to bring and deploy your own model. SAP systems contain a huge amount of enterprise data, which many solutions can analyze. With SAP AI Core, you can use this enterprise data to train machine learning services and help your business applications become intelligent. Your applications learn from historical data that is generated by manually performed tasks, and then uses this new-found knowledge to automate the same tasks in the future.' It also includes tabs for 'Overview', 'Service Plans', 'Documentation', and 'Service Plans'.

3. Enter a name for your service instance and choose [Next](#) (all other details will be filled by default).

New Instance or Subscription

1 Basic Info      2 Parameters      3 Review

Enter basic info for your instance or subscription.

Service: \* [i](#)

SAP AI Core

Plan: \*

standard

Runtime Environment: \*

Cloud Foundry

Space: \*

TestAIFoundation

Instance Name: \* [i](#)

TestAICore|

**Next >**   **Create**   **Cancel**

The 'Instance Name' field is highlighted with a red border.

4. At present, the JSON file upload feature is not supported. Choose [Next](#) to proceed.

## New Instance or Subscription

1 Basic Info      2 Parameters      3 Review

Configure instance parameters. [\(1\)](#)

Upload a JSON file:

Select JSON file

Or specify the parameters in JSON format:  
1

< Back **Next >** Create Cancel

5. Check the data and choose [Create](#).

## New Instance or Subscription

The screenshot shows the third step of a three-step wizard for creating a new instance or subscription. The steps are numbered 1 (Basic Info), 2 (Parameters), and 3 (Review). Step 3 is highlighted with a blue circle and the word 'Review'. The page title is 'New Instance or Subscription'.

**Review and verify the instance details.**

**TestAICore**

**Service:** SAP AI Core  
**Service Plan:** standard  
**Runtime Environment:** Cloud Foundry  
**Space:** TestAIFoundation

**Creating an instance might take a while.**

**Create** (button) **Cancel** (button)

## Results

When your service instance is created, you can view it on the *Instances and Subscriptions* page of your subaccount.

The screenshot shows the 'Instances and Subscriptions' page in the SAP BTP Cockpit. The left sidebar shows navigation options like Overview, Services, Cloud Foundry, and more. The main area displays the subaccount information: 'Subaccount: Testsubaccount - Instances and Subscriptions'. A red box highlights the 'Instances and Subscriptions' link in the sidebar. Another red box highlights the 'Instances (1)' section in the main content area. The table shows one instance: TestAICore, Service: SAP AI Core, Plan: standard, Runtime Environment: Cloud Foundry, Scope: TestAIFoundation, and Status: Created.

Instance	Service	Plan	Runtime Environment	Scope	Credentials	Status
TestAICore	SAP AI Core	standard	Cloud Foundry	TestAIFoundation		Created

## 4.1.6 Create a Service Key

### Procedure

1. On the *Instances and Subscriptions* page, find your new instance and choose *Create Service Key* from the dropdown.

The screenshot shows the SAP BTP Cockpit interface. The left sidebar has a 'Services' section with 'Instances and Subscriptions' highlighted and boxed in red. The main content area is titled 'Subaccount: Testsubaccount - Instances and Subscriptions'. It contains a search bar and filters for 'All Services', 'All Plans', and 'All Statuses'. Below this, there are tabs for 'Subscriptions (0)', 'Instances (1)', and 'Environments (1)'. The 'Instances (1)' tab is selected, showing a table with one row: 'TestAICore' under 'Instance', 'SAP AI Core' under 'Service', 'standard' under 'Plan', 'Cloud Foundry' under 'Runtime Environment', 'TestAIFoundation' under 'Scope', and 'Created' under 'Status'. To the right of the table is a context menu with options: 'Update', 'Create Binding', and 'Create Service Key', with 'Create Service Key' also boxed in red.

2. Enter a name for your service key and click *Create*.

## New Service Key

Service Key Name:\*

 TestAICore

Configure Binding Parameters: [\(i\)](#)

Upload a JSON file:

Or specify the parameters in JSON format:

1

 Create

### → Remember

You are responsible for the rotation of your access credentials and certificates of SAP AI Core within BTP according to regional policy.

3. **Optional:** To use an x.509 certificate instead of client secret credentials, specify the credentials by adjusting and uploading the following JSON code:

```
{  
  "xsuaa": {  
    "credential-type": "x509",  
    "x509": {  
      "key-length": 2048,  
      "validity": 7,  
      "validity-type": "DAYS"  
    }  
  }  
}
```

}

- `key-length`: The byte length of the generated private key. Default: 2048.
  - `validity`: The validity time unit. DAYS, MONTHS and YEARS are supported. Default: DAYS.
  - `validity-type`: The number of time units. Default: 7.

The default combination is a key of length 2048, valid for 7 DAYS.

4. Download your service key to save it.

## Results

You now have your service key, which provides URLs and credentials for accessing the SAP AI Core instance through SAP AI Launchpad, SAP AI Core toolkit, a Third-Party API Platform, or curl.

## → Remember

You are responsible for the rotation of your access credentials and certificates of SAP AI Core within BTP according to regional policy.

## Credentials

aicore-servicekey

Form JSON

```
1 [ {  
2   "clientid": "sb-4376bdb7-b313",  
3   "clientsecret": "xj...b313",  
4   "url": "https://ai....authentication.sap.hana.ondemand.com",  
5   "identityzone": "ai...",  
6   "identityzoneid": "1eb727a0-1",  
7   "appname": "4e!b313",  
8   "serviceurls": {  
9     "AI_API_URL": "https://api.ai.i.ml.hana.ondemand.com"  
10    }  
11  }  
12 }
```

If you have generated a client secret, your key will include:

- `clientid`, `clientsecret`, and `url` can be used to generate your authentication token.
  - `identityzone` and `identityzoneid` represent your tenant ID.
  - `appname` provides the service instance details if service instance isolation is implemented.
  - `serviceurls` allow you to interact with SAP AI Core once your authentication token has been generated.

- `AI_API_URL`: Unified AI API to handle ML artifacts (such as training, data, models, and deployments) across multiple hyperscalers.

If you have generated an x.509 certificate, your key will include:

- `certificate`
- `certurl` can be used to generate your authentication token.
- `key` your RSA private key.
- `identityzone` and `identityzoneid` represent your tenant ID.
- `appname` provides the service instance details if service instance isolation is implemented.
- `serviceurls` allow you to interact with SAP AI Core once your authentication token has been generated.
  - `AI_API_URL`: Unified AI API to handle ML artifacts (such as training, data, models, and deployments) across multiple hyperscalers.

## 4.1.7 Use a Service Key

After you have created your service key, it can be used by local clients, apps in other spaces, or entities outside your deployment to access SAP AI Core through one of the available interfaces.

## Using a Third-Party API Platform

### Prerequisites

- You have downloaded and installed the API platform of your choice.
- You have familiarized yourself with the documentation and interface of the platform.

### Procedure

1. Download the JSON collection from [https://api.sap.com/api/AI\\_CORE\\_API/overview](https://api.sap.com/api/AI_CORE_API/overview).
2. Import the JSON file to the API platform.
3. After the import is complete, highlight the collection and select the *Authorization* tab.
4. Navigate to *Configure New Token*, enter the credentials from your service key, and save your changes.

#### ⓘ Note

- The *Token Name* field is your choice of descriptive identifier.
- The *Access Token URL* is labeled *url* in your service key. Add `/oauth/token` to the end of the URL.
- The *Grant Type* should be `Client Credentials`.

If you see an alert relating to the characters in your credentials, ignore it.

### ⓘ Note

If you have generated a x.509 certificate instead of client secret credentials, you'll need to use your `certificate`, `key` and `certUrl` to create your token.

For example: `curl --cert <cert.pem> --key <key.pem> -XPOST <certUrl>/oauth/token -d 'grant_type=client_credentials&client_id=<client id>'`

5. Select the *Variables* tab, and set your `baseUrl` from your credentials.

The `baseUrl` is labeled `AI_API_URL` in your service key.

6. Choose *Save*.
7. In the *Authorization* tab, choose *Get New Access Token*. and wait for the authentication process. When the authentication process is complete, select *Use Token* to finish. Check that the token is stored in your environment variables. If it is not stored automatically, you can copy and paste it to the `Token` field manually.

## Next Steps

To train and deploy your own AI models, follow the procedure in [Administration \[page 80\]](#).

To use generative AI models provided in the generative AI hub, see [Generative AI Hub in SAP AI Core \[page 125\]](#).

## Using Curl

### Prerequisites

curl is likely to be installed on your operating system by default. To check, open a command prompt and enter `curl -v`. If curl isn't installed, download and install it from <https://curl.se/>.

### ⓘ Note

On macOS, you may need to install `jq` so that you can follow the curl commands.

1. Install brew from <https://brew.sh/>.
2. In a Terminal session, run `brew install jq` to install `jq` in your shell environment.

### Procedure

1. Set up your environment as follows:

**For Linux:**

```
# XSUAA details
```

```
# URLs should be without trailing slash '/'
export CLIENTID=<clientid> from service key
export CLIENTSECRET=<clientsecret> from service key
export XSUAA_URL=<url> from service key
export AI_API_URL=<AI_API_URL> from service key
```

#### For Windows PowerShell:

```
$env:CLIENTID = <clientid> from service key
$env:CLIENTSECRET = <clientsecret> from service key
$env:XSUAA_URL = <url> from service key
$env:AI_API_URL = <AI_API_URL> from service key
```

#### ⓘ Note

The `export` command sets the values of your keys to your environment variable, meaning that they will be retained after you close your terminal session. It is possible to set the environment variables without the `export`, for the current session only.

- Get the XSUAA OAuth Token using `clientid` and `clientsecret` from the service key to call the APIs.

The XSUAA OAuth token is required for authentication when making AI API calls.

#### For Linux:

```
SECRET=`echo -n '$CLIENTID:$CLIENTSECRET' | base64 -i -` 
TOKEN=`curl --location --request POST "$XSUAA_URL/oauth/token?
grant_type=client_credentials"
--header "Authorization: Basic $SECRET" | jq -r '.access_token'`
```

#### For Windows PowerShell:

```
$SECRET = $env:CLIENTID + ":" + $env:CLIENTSECRET
$base64SECRET =
[Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($SECRET))
$TOKENRESPONSE = Invoke-WebRequest -Method Post "$env:XSUAA_URL/oauth/token"
-Headers @{
    "Authorization" = "Basic $base64SECRET";
    "Content-Type" =
    "application/x-www-form-urlencoded"
} -Body "grant_type=client_credentials"
$TOKENJSON = $TOKENRESPONSE.Content | ConvertFrom-Json
$TOKEN = $TOKENJSON.access_token
```

#### ⓘ Note

The token is valid for a limited time. Once it has expired, create a new token, using the same code snippet.

#### ⓘ Note

If you have generated a x.509 certificate instead of client secret credentials, you'll need to use your `certificate`, `key` and `certUrl` to create your token.

For example: `curl --cert <cert.pem> --key <key.pem> -XPOST <certUrl>/oauth/token -d 'grant_type=client_credentials&client_id=<client id>'`

- Verify that the token has been fetched properly:

```
echo $TOKEN
```

You should see a long string of alphanumeric characters:

```
eyJhbGciOiJSUzI1NiIsImprdSI6Imh0dHBzOi8vYWktYWxwaGETdmFsaWRhdGlvbi0yLmF1dGh1bnRpY2F0aW9uLnNhcC5oYW5hLm9uZGVtYW5kLmNvbS90b2tlbl9rZXlziwiia2lkIjoizGVmYXVsdC1qd3Qta2V5LTMyODMzMjg2NCIsInR5cCI6IkpxVCJ94ZGU5YjAxNmQ0MDk5YjlmM...  
.....ALdfbMsHoYTtF6fNFbf3ZQ
```

## Next Steps

To train and deploy your own AI models, follow the procedure in [Administration \[page 80\]](#).

To use generative AI models provided in the generative AI hub, see [Generative AI Hub in SAP AI Core \[page 125\]](#).

## 4.2 Enabling the Service in the Kyma Environment

Enable SAP AI Core using the standard procedures for the SAP BTP Kyma environment.

### Procedure

1. Create a service instance in the Kyma environment.
2. You can then bind the service instance to your application, or you can create a service key to communicate directly with the service instance. See [Using SAP BTP Services in the Kyma Environment](#).

### Related Information

[Using SAP BTP Services in the Kyma Environment](#)

# 5 SAP AI Core Starter Tutorials

You can use a booster to get started with the free plans for SAP AI Core and SAP AI Launchpad using the tutorial: [AI Boosters](#).

Alternatively, to complete the booster tutorial, then learn the basics of SAP AI Core from beginning to end in a simple use case, follow the tutorial: [Quick Start for SAP AI Core](#). If you have already provisioned SAP AI Core, you can skip through to the second tutorial in this guide.

To see the complete library of tutorials available for SAP AI Core, see [Tutorials \[page 576\]](#).

## ⓘ Note

Once you have provisioned SAP AI Core, you can access the service in multiple ways. Where the tutorial below offers more than one option tab, you need only complete the tab for your chosen method of access. For customers who prefer a GUI to code, SAP AI Launchpad is recommended. For more information, see [SAP AI Launchpad](#).

# 6 Administration

Creating secrets for external programs and tools, that are used with SAP AI Core means that you can connect them without compromising your credentials.

Using SAP AI Core alongside external tools such as GitHub, Docker and Amazon Web Services S3 storage leverages the benefits of version control, containerization, and cloud storage. Your content is made available remotely, if you have a stable internet connection.

Administration is a one-time procedure, but steps can be repeated, if necessary, for example to remove or add a tool.

## ⓘ Note

You must have completed the initial setup tasks before configuring your SAP AI Core instance. For more information, see [Initial Setup \[page 57\]](#).

### [Manage Resource Groups \[page 90\]](#)

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.

## 6.1 Manage Your Git Repository

### 6.1.1 Add a Git Repository

You can use your own git repository to version control your SAP AI Core templates. The GitOps onboarding to SAP AI Core instances involves setting up your git repository and synchronizing your content.

## Using Curl

### Prerequisites

- 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.

- You've completed the initial setup. For more information, see [Initial Setup \[page 57\]](#).

### ⓘ 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.

## Context

Git repositories are managed by creating personal access tokens and registering 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 `{apiurl}/v2/admin/repositories` and include your credentials:

```
curl --location --request POST "$AI_API_URL/v2/admin/repositories" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--data-raw '{
    "url": "https://github.com/john/examplerrepo",
    "username": "john",
    "password": "<GIT_PAT_USER_TOKEN>"
}'
```

You specify your unique git repository details as follows:

- `url`: URL of the git repository
- `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, register the repository in SAP AI Core separately for each tenant and provide the **same** `username` and `password`.

## Next Steps

Create an application to sync your folders. For more information, see [Create an Application \[page 86\]](#).

# Using a Third-Party API Platform

## Prerequisites

- 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.
- You've completed the initial setup. For more information, see [Initial Setup \[page 57\]](#).

### ⓘ 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.

## Context

Git repositories are managed by creating personal access tokens and registering 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

Send a POST request to the endpoint `{ {apiurl} }/v2/admin/repositories` and include your credentials in JSON format in the `raw` body:

You specify your unique git repository details as follows:

- `url`: URL of the git repository

- **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, register the repository in SAP AI Core separately for each tenant and provide the **same** **username** and **password**.

## Next Steps

Create an application to sync your folders. For more information, see [Create an Application \[page 86\]](#).

### 6.1.2 Edit a Git Repository

#### Using Curl

##### Procedure

Submit a PATCH request to the endpoint `{apiurl}/v2/admin/repositories` and include your changes:

```
curl --location --request PATCH "$AI_API_URL/v2/admin/repositories" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--data-raw '{
    "url": "https://github.com/john/examplerrepo",
    "username": "john",
    "password": "<GIT_PAT_USER_TOKEN>"
}'
```

You specify your unique git repository details as follows:

- **url**: URL of the git repository
- **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, register the repository in SAP AI Core separately for each tenant and provide the **same** **username** and **password**.

## Using a Third-Party API Platform

### Procedure

Send a PATCH request to the endpoint {{apiurl}}/v2/admin/repositories/{{repositoryName}} and include your changes in the body.

You specify your unique git repository details as follows:

- `url`: URL of the git repository
- `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, register the repository in SAP AI Core separately for each tenant and provide the **same** username and password.

### 6.1.3 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 repo 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 repo 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.

## 6.2 Manage Applications

### 6.2.1 Create an Application

#### Using Curl

##### Context

After registering your Git repository, create an application to sync the templates in your repository. The first sync takes some time. You can check the application's status to see when it's complete. After the initial sync, syncing occurs automatically approximately every three minutes. You can also request it manually.

##### ⓘ Note

Do not create applications that attempt to sync the same source. If two apps have the same `repositoryURL`, `revision`, and `path`, syncing will fail.

#### Procedure

Submit a POST request to the endpoint `{ {apiurl} }/v2/admin/applications` including details of your application:

```
curl --location --request POST "$AI_API_URL/v2/admin/applications" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--data-raw '{
    "applicationName": "my-app",
    "repositoryUrl": "https://github.com/john/examplerrepo",
    "revision": "HEAD",
    "path": "workflows"
}'
```

- `applicationName`: Set a name for your application. The name must be between 3 and 64 characters long and match `[A-Za-z0-9\-\_]+`.
- `repositoryUrl`: The URL of a registered git repository. The URL is case-sensitive and has to match the URL of a registered git repository.
- `revision`: The revision to target. `<HEAD>` refers to the most recent.
- `path`: The path to the target folder that contains the templates to be synced.

Because each application points to a particular path and revision in the repository, multiple applications can be created for the same `repositoryUrl`.

## Results

After the GitOps setup is completed, the templates in your git repository are automatically synced to SAP AI Core Approximately every three minutes.

## Next Steps

Check the synchronization status of your application by submitting a GET request to {{apiurl}}/v2/admin/applications/{{appName}}/status:

```
curl --location --request GET "$AI_API_URL/v2/admin/applications/{{appName}}/status" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json'
```

As applicationName, enter the name of your application that you specified when you created the application.

## Using a Third-Party API Platform

### Context

After registering your Git repository, create an application to sync the templates in your repository. The first sync takes some time. You can check the application's status to see when it's complete. After the initial sync, syncing occurs automatically approximately every three minutes. You can also request it manually.

#### ⓘ Note

Do not create applications that attempt to sync the same source. If two apps have the same repositoryURL, revision, and path, syncing will fail.

### Procedure

Send a POST request to the endpoint {{apiurl}}/v2/admin/applications including details of your application:

- **applicationName:** Set a name for your application. The name must be between 3 and 64 characters long and match **[A-Za-z0-9\-\\_]+**.
- **repositoryUrl:** The URL of a registered git repository. The URL is case-sensitive and has to match the URL of a registered git repository.
- **revision:** The revision to target. **<HEAD>** refers to the most recent.
- **path:** The path to the target folder that contains the templates to be synced.

Because each application points to a particular path and revision in the repository, multiple applications can be created for the same `repositoryUrl`.

## Results

After the GitOps setup is completed, the templates in your git repository are automatically synced to SAP AI Core Approximately every three minutes.

## Next Steps

Check the synchronization status of your application by sending a GET request to `{apiurl}/v2/admin/applications/{appName}/status`. As `applicationName`, enter the name of your application that you specified when you created the application.

### ↔ Output Code

```
{
  "healthStatus": "Healthy",
  "message": "successfully synced (all tasks run)",
  "reconciledAt": "2021-11-23T10:27:49Z",
  "source": {
    "path": "workflows",
    "repoURL": "https://github.com/username/examplerrepo",
    "revision": "db611bb28be3c853d08867c08b52b8f733b4f7bf",
  },
  "syncFinishedAt": "2021-11-23T10:27:49Z",
  "syncResourceStatus": [
    {
      "kind": "ServingTemplate",
      "message": "servingtemplate.ai.sap.com/text-clf-infer-tutorial configured",
      "name": "text-clf-infer-tutorial",
      "status": "Synced",
    }
  ],
  "syncedStartedAt": "2021-11-23T10:27:48Z",
  "syncStatus": "Synced",
}
```

## Sync an Application Manually

Applications sync with your GitHub repository automatically at intervals of ~3 minutes. Use the endpoint below to manually request a sync:`{apiurl}/admin/applications/{appName}/refresh`

## 6.2.2 List Applications

### Using Curl

#### Procedure

Send a GET request to the endpoint {{apiurl}}/v2/admin/applications.

### Using a Third-Party API Platform

#### Procedure

Send a GET request to the endpoint {{apiurl}}/v2/admin/applications.

## 6.2.3 Edit an Application

### Using Curl

#### Procedure

Send a PATCH request to the endpoint {{apiurl}}/v2/admin/applications/{{appName}} and include your changes in the body.

### Using a Third-Party API Platform

#### Procedure

Send a PATCH request to the endpoint {{apiurl}}/v2/admin/applications/{{appName}} and include your changes in the body.

## 6.2.4 Delete an Application

### Using Curl

#### Procedure

Send a DELETE request to the endpoint {{apiurl}}/v2/admin/applications/{{appName}}.

### Using a Third-Party API Platform

#### Procedure

Send a DELETE request to the endpoint {{apiurl}}/v2/admin/applications/{{appName}}.

## 6.3 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.

## ⚠ Restriction

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 \[page 95\]](#).

[Create a Resource Group \[page 92\]](#)

[Edit a Resource Group \[page 93\]](#)

[Delete a Resource Group \[page 95\]](#)

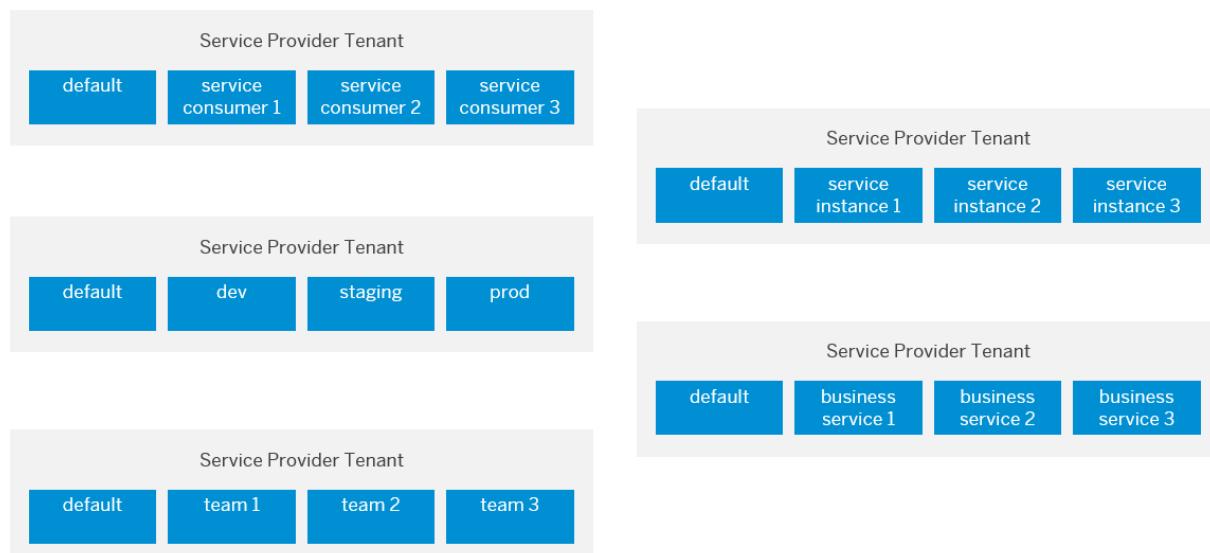
**Parent topic:** Administration [\[page 80\]](#)

## 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:



## 6.3.1 Create a Resource Group

**Parent topic:** Manage Resource Groups [page 90]

### Related Information

[Edit a Resource Group \[page 93\]](#)

[Delete a Resource Group \[page 95\]](#)

## Using Curl

### Prerequisites

You've completed the initial setup. For more information, see [Initial Setup \[page 57\]](#).

You have access to a public-facing Docker registry over the internet. It isn't possible to use a Docker registry behind a VPN or corporate network.

### 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

Create a resource group by sending the following:

```
curl --location --request POST "$AI_API_URL/v2/admin/resourceGroups" --header "Authorization: Bearer $TOKEN" --header 'Content-Type: application/json' --data-raw '{ \"resourceGroupId\": "<ID of your resource group>" }'
```

# Using a Third-Party API Platform

## Prerequisites

You've completed the initial setup. For more information, see [Initial Setup \[page 57\]](#).

You have access to a public-facing Docker registry over the internet. It isn't possible to use a Docker registry behind a VPN or corporate network.

## 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

- As the request body, select the `raw` radio button and enter the following:

```
{  
    "resourceGroupId": "<ID of your resource group>"  
}
```

- Send the request.

## Results

You'll receive a 202 response to confirm that the request to create the resource group has been accepted.

## 6.3.2 Edit a Resource Group

**Parent topic:** [Manage Resource Groups \[page 90\]](#)

## Related Information

[Create a Resource Group \[page 92\]](#)

[Delete a Resource Group \[page 95\]](#)

## Using Curl

### 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

Create 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>" }'
```

## 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>"  
}
```

### 6.3.3 Delete a Resource Group

**Parent topic:** [Manage Resource Groups \[page 90\]](#)

## Related Information

[Create a Resource Group \[page 92\]](#)

[Edit a Resource Group \[page 93\]](#)

## Using Curl

### 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>" }'
```

## Using a Third-Party API Platform

### Procedure

Send a DELETE request to the endpoint {{apiurl}}/v2/admin/resourceGroups/{{resource\_group\_name}}.

## 6.4 Manage Object Store Secrets

### 6.4.1 Register an Object Store Secret

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 SAP AI Core.

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.

#### ⚠ Restriction

You must create an **object store secret** named **default** to store the training output artifact (for example, a model). If this default object store secret is missing, the training pipeline fails.

For **input training artifacts only**, you can create multiple object store secrets with different names as needed.

#### → Remember

You are responsible for the rotation of your access credentials and certificates of SAP AI Core within BTP according to regional policy.

## Using Curl

### 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.
- SAP AI Core supports multiple hyperscaler object stores, including the following:
  - Amazon S3
  - Azure Blob Storage
  - Google Cloud Storage
  - OSS (Alibaba Cloud Object Storage Service)
  - SAP HANA Cloud, Data Lake

### Procedure

Register your object store secret details using the endpoint `/v2/admin/objectStoreSecrets`.

## ⓘ 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": "xxxxxx",
        "SECRET_ACCESS_KEY": "xxxxxx"
    }
}'
```

- For SAP HANA Cloud, Data Lake:

```
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": "webhdfs",
    "pathPrefix": "<path prefix to be appended>",
    "data": {
        // e.g. https://c32727c8-4260-4c37-b97f-ede322dcfa8f.files.hdl.canary-eu10.hanacloud.ondemand.com
        "HDFS_NAMENODE": "https://<file-container-name>.files.hdl.canary-eu10.hanacloud.ondemand.com",
        "TLS_CERT": "-----BEGIN CERTIFICATE-----\nMIICmjCCAYIxXXXXXXXXXXXXR4wtC32bGO66D+Jc8RhaIA==\n-----END CERTIFICATE-----\n",
        "TLS_KEY": "-----BEGIN PRIVATE KEY-----\nMIIEvQIBADANBgkqXXXXXXXXXXnor+rtZHhzEfX5dYLCS5Pww=\n-----END PRIVATE KEY-----\n",
    }
}'
```

```
    "HEADERS": "{\"x-sap-filecontainer\": \"<file-container-name>\",  
    \"Content-Type\": \"application/octet-stream\"}"}  
}'
```

### ⚠ Restriction

When using an SAP HANA Data Lake object store with output artifacts pointing to a directory, you can't use `archive: none: {}` in your workflow templates to disable artifact archiving. For more information, see [Workflow Templates \[page 464\]](#).

- 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  
    }  
}'
```

- For Google Cloud Storage (GCS):

```
curl --location --request PATCH "$AI_API_URL/v2/admin/objectStoreSecrets/  
{objectStoreName}" \  
--header "Authorization: Bearer $TOKEN" \  
--header 'Content-Type: application/json' \  
--header 'AI-Resource-Group: <Resource group>' \  
--data-raw '{  
    "name": "default",  
    "type": "gcs",  
    "pathPrefix": "<path prefix to be appended>",  
    "data": {  
        "BUCKET": "<gcs bucket name>", //  
        required  
        "PRIVATE_KEY": "<base64 encoded service account key>", //  
        required  
    }  
}'
```

### → 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.

# Using a Third-Party API Platform

## Prerequisites

You've completed the initial setup. For more information, see [Initial Setup \[page 57\]](#).

You have access to a public-facing Docker registry over the internet. It isn't possible to use a Docker registry behind a VPN or corporate network.

## 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.
- SAP AI Core supports multiple hyperscaler object stores, including the following:
  - Amazon S3
  - Azure Blob Storage
  - Google Cloud Storage
  - OSS (Alibaba Cloud Object Storage Service)
  - SAP HANA Cloud, Data Lake

## Procedure

1. Send a POST request to the endpoint `{apiurl}/v2/admin/objectStoreSecrets`.
2. As the request body, select the `raw` radio button and enter your object store secret details.

### ⓘ Note

For all storage types **except** Azure Blob Storage, all `<data>` fields are required. For Azure, required fields are specified.

- For Amazon S3:

```
{
    "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):

```
{
```

```

    "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": "xxxxxx",
        "SECRET_ACCESS_KEY": "xxxxxx"
    }
}

```

- For SAP HANA Cloud, Data Lake:

```

{
    "name": "default",
    "type": "webhdfs",
    "pathPrefix": "<path prefix to be appended>",
    "data": {
        // e.g. https://c32727c8-4260-4c37-b97f-ede322dcfa8f.files.hdl.canary-eu10.hanacloud.ondemand.com
        "HDFS_NAMENODE": "https://<file-container-name>.files.hdl.canary-eu10.hanacloud.ondemand.com",
        "TLS_CERT": "-----BEGIN CERTIFICATE-----\nMIICmjCCAYIxXXXXXXXXXXXXX4wtC32bGO66D+Jc8RhaIA==\n-----END CERTIFICATE-----\n",
        "TLS_KEY": "-----BEGIN PRIVATE KEY-----\nMIIEvQIBADANBgkqXXXXXXXXXXXXnor+rtZHhzEfx5dYLC5Pww=\n-----END PRIVATE KEY-----\n",
        "HEADERS": "{\"x-sap-filecontainer\": \"<file-container-name>\",\n\"Content-Type\": \"application/octet-stream\"}"
    }
}

```

## ⚠ Restriction

When using an SAP HANA Data Lake object store with output artifacts pointing to a directory, you can't use `archive: none: {}` in your workflow templates to disable artifact archiving. For more information, see [Workflow Templates \[page 464\]](#).

- For Azure Blob Storage:

```

{
    "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
    }
}

{
    "name": "default",
    "type": "gcs",
    "pathPrefix": "<path prefix to be appended>",
    "data": {
        "BUCKET": "<gcs bucket name>", // required
    }
}

```

```
        "PRIVATE_KEY": "<base64 encoded service account key>", //  
    required  
}  
}
```

#### → 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.

3. Send the request.

## 6.4.2 Edit an Object Store Secret

### Using Curl

#### 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.
- SAP AI Core supports multiple hyperscaler object stores, including the following:
  - Amazon S3
  - Azure Blob Storage
  - Google Cloud Storage
  - OSS (Alibaba Cloud Object Storage Service)
  - SAP HANA Cloud, Data Lake

### Procedure

Edit your object store secret details using the endpoint `$AI_API_URL/v2/admin/objectStoreSecrets/{objectStoreName}`.

#### ⓘ 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 PATCH "$AI_API_URL/v2/admin/objectStoreSecrets/
{{objectStoreName}}" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <Resource group>' \
--data-binary '{
    "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 PATCH "$AI_API_URL/v2/admin/objectStoreSecrets/
{{objectStoreName}}" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <Resource group>' \
--data-binary '{
    "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": "xxxxxx",
        "SECRET_ACCESS_KEY": "xxxxxx"
    }
}'
```

- For SAP HANA Cloud, Data Lake:

```
curl --location --request PATCH "$AI_API_URL/v2/admin/objectStoreSecrets/
{{objectStoreName}}" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <Resource group>' \
--data-binary '{
    "name": "default",
    "type": "webhdfs",
    "pathPrefix": "<path prefix to be appended>",
    "data": {
        // e.g. https://c32727c8-4260-4c37-b97f-ed322dcfa8f.files.hdl.canary-
        eu10.hanacloud.ondemand.com
        "HDFS_NAMENODE": "https://<file-container-name>.files.hdl.canary-
        eu10.hanacloud.ondemand.com",
        "TLS_CERT": "-----BEGIN CERTIFICATE-----
\nMIICmjCCAYIxXXXXXXXXXXXXX4wtC32bG066D+Jc8RhaIA==\n-----END CERTIFICATE-----
\n",
        "TLS_KEY": "-----BEGIN PRIVATE KEY-----
\nMIIEvQIBADANBgkqXXXXXXXXXXnor+rtZHhzEfX5dYLCS5Pww=\n-----END PRIVATE
KEY-----\n",
        "HEADERS": "{\"x-sap-filecontainer\": \"<file-container-name>\",
        \"Content-Type\": \"application/octet-stream\"}"
    }
}'
```

- For Azure Blob Storage:

```
curl --location --request PATCH "$AI_API_URL/v2/admin/objectStoreSecrets/
{{objectStoreName}}" \
--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
    }
}'
```

- For Google Cloud Storage (GCS):

```
curl --location --request PATCH "$AI_API_URL/v2/admin/objectStoreSecrets/
{{objectStoreName}}" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: <Resource group>' \
--data-raw '{
    "name": "default",
    "type": "gcs",
    "pathPrefix": "<path prefix to be appended>",
    "data": {
        "BUCKET": "<gcs bucket name>", //
        required
        "PRIVATE_KEY": "<base64 encoded service account key>", //
        required
    }
}'
```

#### → 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.

#### ① Note

If the `AI-Resource-Group` header isn't specified, the `<Resource Group>` is assigned the value "default" automatically.

# Using a Third-Party API Platform

## 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.

SAP AI Core supports multiple hyperscaler object stores, including the following:

- Amazon S3
- Azure Blob Storage
- Google Cloud Storage
- OSS (Alibaba Cloud Object Storage Service)
- SAP HANA Cloud, Data Lake

## Procedure

1. Send a PATCH request to the endpoint .
2. As the request body, select the [raw](#) radio button and enter your object store secret details.

### ⓘ Note

For all storage types **except** Azure Blob Storage, all `<data>` fields are required. For Azure, required fields are specified.

- For Amazon S3:

```
{  
    "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):

```
{  
    "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": "xxxxxx",  
        "SECRET_ACCESS_KEY": "xxxxxx"  
    }  
}
```

- For SAP HANA Cloud, Data Lake:

```
{
  "name": "default",
  "type": "webhdfs",
  "pathPrefix": "<path prefix to be appended>",
  "data": {
    // e.g. https://c32727c8-4260-4c37-b97f-ed322dcfa8f.files.hdl.canary-
    eu10.hanacloud.ondemand.com
    "HDFS_NAMENODE": "https://<file-container-name>.files.hdl.canary-
    eu10.hanacloud.ondemand.com",
    "TLS_CERT": "-----BEGIN CERTIFICATE-----
    \nMIICMjCCAYIxXXXXXXXXXXXX4wtC32bGO66D+Jc8RhaIA==\n-----END
    CERTIFICATE-----\n",
    "TLS_KEY": "-----BEGIN PRIVATE KEY-----
    \nMIIEvQIBADANBgkqXXXXXXXXXXXXnor+rtZHhzEfx5dYLC5Pww=\n-----END PRIVATE
    KEY-----\n",
    "HEADERS": "{\"x-sap-filecontainer\": \"<file-container-name>\",
    \"Content-Type\": \"application/octet-stream\"}"
  }
}
```

- For Azure Blob Storage:

```
{
  "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
  }
}

{
  "name": "default",
  "type": "gcs",
  "pathPrefix": "<path prefix to be appended>",
  "data": {
    required
    "BUCKET": "<gcs bucket name>", //required
    required
    "PRIVATE_KEY": "<base64 encoded service account key>", //required
  }
}
```

### → 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.

3. Send the request.

### 6.4.3 Delete an Object Store Secret

#### Using Curl

##### Context

Deleting an object store secret stops access to the object store.

##### Procedure

Run the following code:

```
curl --location --request DELETE "$AI_API_URL/v2/admin/objectStoreSecrets/\\{\\{objectStoreName}\\}\\\" \\
```

#### Using a Third-Party API Platform

##### Context

Deleting an object store secret stops access to the object store.

##### Procedure

Send a DELETE request to the endpoint

## 6.5 Manage Docker Registry Secrets

### 6.5.1 Register Your Docker Registry Secret

Docker packages and runs applications in remote containers. Connect SAP AI Core to a Docker repository and manage access using a Docker registry secret.

## Using Curl

### Prerequisites

You've completed the initial setup. For more information, see [Initial Setup \[page 57\]](#).

You have access to a public-facing Docker registry over the internet. It isn't possible to use a Docker registry behind a VPN or corporate network.

### Context

Your Docker credentials are managed using secrets. Secrets allow and control connections across directories and tools 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.

Your Docker registry secret lets you authorize SAP AI Core to pull your **private** Docker images from your Docker repository. You specify the name of the secret in your workflows to authenticate the Docker image pull. For more information, see [Workflow Templates \[page 464\]](#) and [Serving Templates \[page 513\]](#).

### Procedure

1. Submit a POST request to the endpoint `{apiurl}/v2/admin/dockerRegistrySecrets`. Include the following parameters in your request body:
  - `name`: Set the name of your Docker registry secret. This is your choice of identifier for your secret. In the example, the name is "mydockerregistry".
  - `data`: Enter a JSON string that represents your Docker registry secret.

#### ↔ Sample Code

```
{
```

```
"name": "mydockerregistry",
"data": {
  ".dockerconfigjson": "{\"auths\":{\"your.private.registry\":
  {\"username\":\"john\", \"password\":\"docker-accesstoken-or-password\"}}}"
}
```

### ⓘ Note

If you are using a public Docker registry from <http://hub.docker.com>, you must provide your Docker URL in the format `https://index.docker.io`, in the `<"auths\>` variable input.

### ⚡ Sample Code

```
$ curl --location --request POST "$AI_API_URL/v2/admin/
dockerRegistrySecrets" --header "Authorization: Bearer $TOKEN" --header
'Content-Type: application/json' --data-raw '{
  "name": "mydockerregistry",
  "data": {
    ".dockerconfigjson": "{\"auths\":{\"my.docker.repositories.io\":
    {\"username\":\"$USERNAME\", \"password\":\"$PWD\"}}}"
  }
}
{
  "message": "secret has been created"
}'
```

- After your Docker registry secret has been created, reference it in your template as an image pull secret. For example

### ⚡ Source Code

```
spec:
  imagePullSecrets:
    - name: <Name of your Docker registry secret>
```

## Using a Third-Party API Platform

### Prerequisites

You've completed the initial setup. For more information, see [Initial Setup \[page 57\]](#).

You have access to a public-facing Docker registry over the internet. It isn't possible to use a Docker registry behind a VPN or corporate network.

### Context

Your Docker credentials are managed using secrets. Secrets allow and control connections across directories and tools without compromising your credentials.

Your Docker registry secret lets you authorize SAP AI Core to pull your **private** Docker images from your Docker repository. You specify the name of the secret in your workflows to authenticate the Docker image pull. For more information, see [Workflow Templates \[page 464\]](#) and [Serving Templates \[page 513\]](#).

## Procedure

1. Send a POST request to the endpoint `{ {apiurl} }/v2/admin/dockerRegistrySecrets`
2. As the request body, select the `raw` radio button and enter the following:

```
{  
  "name": "mydockerregistry",  
  "data": {  
    ".dockerconfigjson": " {\"auths\":{\"your.private.registry\":  
      {\"username\":\"john\", \"password\":\"docker-accesstoken-or-password\"}}}"  
  }  
}
```

- **name:** Set the name of your Docker registry secret. This is your choice of identifier for your secret. In the example, the name is "mydockerregistry".
- **data:** Enter a JSON string that represents your Docker registry secret.

3. Send the request:

### ↔ Output Code

```
{  
  "message": "secret has been created"  
}
```

4. After your Docker registry secret has been created, reference it in your template as an image pull secret.

```
spec:  
  imagePullSecrets:  
    - name: <Name of your Docker registry secret>
```

## 6.5.2 Edit a Docker Registry Secret

Docker packages and runs applications in remote containers. Connect SAP AI Core to a Docker repository and manage access using a Docker registry secret.

## Using Curl

### Context

Your Docker credentials are managed using secrets. Secrets allow and control connections across directories and tools without compromising your credentials.

Your Docker registry secret lets you authorize SAP AI Core to pull your **private** Docker images from your Docker repository. You specify the name of the secret in your workflows to authenticate the Docker image pull. For more information, see [Workflow Templates \[page 464\]](#) and [Serving Templates \[page 513\]](#).

## Procedure

Submit a PATCH request to the endpoint `$AI_API_URL/v2/admin/dockerRegistrySecrets/{dockerRegistryName}`. Include the following parameters in your request body:

- **name:** Set the name of your Docker registry secret.
- **data:** Enter a JSON string that represents your Docker registry secret.

### Sample Code

```
{  
  "name": "mydockerregistry",  
  "data": {  
    ".dockerconfigjson": "{\"auths\":{\"your.private.registry\"::  
      {\"username\":\"john\", \"password\":\"docker-accesstoken-or-password\"}}}"  
  }  
}
```

### Note

If you are using a public Docker registry from <http://hub.docker.com>, you must provide your Docker URL in the format `https://index.docker.io`, in the `<\\"auths\\>` variable input.

```
$ curl --location --request PATCH "$AI_API_URL/v2/admin/dockerRegistrySecrets/  
{dockerRegistryName}" --header "Authorization: Bearer $TOKEN" --header  
'Content-Type: application/json' --data-raw '{  
  "name": "mydockerregistry",  
  "data": {  
    ".dockerconfigjson": "{\"auths\": {\"my.docker.repositories.io\":  
      {\"username\": \"$USERNAME\", \"password\": \"$PWD\"}}}"  
  }  
}'
```

## Using a Third-Party API Platform

## Procedure

Send a PATCH request to the endpoint `{apiurl}/v2/admin/repositories/{repositoryName}` and include your changes in the body.

You specify your unique git repository details as follows:

- **url:** URL of the git repository
- **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, register the repository in SAP AI Core separately for each tenant and provide the **same** username and password.

### 6.5.3 Delete a Docker Registry Secret

Deleting a docker registry secret removes access to the docker registry.

#### Using Curl

##### Procedure

Submit a DELETE request to the endpoint `$AI_API_URL/v2/admin/dockerRegistrySecrets/{ {dockerRegistryName} }`.

#### Using a Third-Party API Platform

##### Procedure

Send a DELETE request to the endpoint `{ {apiurl}}/v2/admin/dockerRegistrySecrets/{ {dockerRegistryName} }`

## 6.6 Manage Generic Secrets

### 6.6.1 Create a Generic Secret

A generic secret authorizes SAP AI Core to use your resource group without exposing your credentials.

#### Using Curl

##### Prerequisites

You've completed the initial setup. For more information, see [Initial Setup \[page 57\]](#).

You have access to a public-facing Docker registry over the internet. It isn't possible to use a Docker registry behind a VPN or corporate network.

## 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.

### → Remember

You are responsible for the rotation of your access credentials and certificates of SAP AI Core within BTP according to regional policy.

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](#).

Each tenant can have a maximum of five tenant-wide secrets. If you reach this limit, you receive an error message. To free up space, delete tenant-wide secrets as described at [Delete a Generic Secret \[page 122\]](#). Alternatively, submit a ticket to request an increase in your quota.

### → Tip

Generic secrets created at the tenant level automatically propagate to all resource groups. However, if a generic secret with the same name is created at the resource-group level, it replaces the tenant-level secret at the time of creation. The system periodically overrides resource-group level secrets with the corresponding tenant-level secret, but this process can take some time. If a resource-group user creates a secret with the same name as an existing tenant-wide secret, it temporarily overwrites the tenant-wide secret at the resource-group level. This behavior can cause issues, especially for critical operations such as metering.

To prevent unintended overwrites, ensure that the tenant prevents resource-group users from creating arbitrary secrets. You can do so in the following ways:

- Restrict users at resource-group level from accessing the `secrets` endpoint by withholding the JWT token.
- Allow users at resource-group level to create generic secrets by making a request using a different authentication mechanism. The main tenant can then validate and transform these requests before

propagating them to the runtime adapter, ensuring that secret names remain consistent and critical secrets aren't unintentionally modified.

## Procedure

Send a POST request and enter the URL `{apiurl}/v2/admin/secrets`.

- **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 POST "$AI_API_URL/v2/admin/secrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'Content-Type: application/json' \
--header 'AI-Resource-Group: default' \
--data-raw '{
    "name": "MY_GENERIC_SECRET",
    "data": {
        "some-credential": "bXktc2Vuc2l0aXZlLWRhdGE="
    }
}'
```

### ⓘ Note

The secret name is written without hyphens to make it simple to consume as a Unix environment variable later. It is written in capitals as is convention.

# Using a Third-Party API Platform

## Prerequisites

You've completed the initial setup. For more information, see [Initial Setup \[page 57\]](#).

You have access to a public-facing Docker registry over the internet. It isn't possible to use a Docker registry behind a VPN or corporate network.

## 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.

### → Remember

You are responsible for the rotation of your access credentials and certificates of SAP AI Core within BTP according to regional policy.

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](#).

Each tenant can have a maximum of five tenant-wide secrets. If you reach this limit, you receive an error message. To free up space, delete tenant-wide secrets as described at [Delete a Generic Secret \[page 122\]](#). Alternatively, submit a ticket to request an increase in your quota.

### → Tip

Generic secrets created at the tenant level automatically propagate to all resource groups. However, if a generic secret with the same name is created at the resource-group level, it replaces the tenant-level secret at the time of creation. The system periodically overrides resource-group level secrets with the corresponding tenant-level secret, but this process can take some time. If a resource-group user creates a secret with the same name as an existing tenant-wide secret, it temporarily overwrites the tenant-wide secret at the resource-group level. This behavior can cause issues, especially for critical operations such as metering.

To prevent unintended overwrites, ensure that the tenant prevents resource-group users from creating arbitrary secrets. You can do so in the following ways:

- Restrict users at resource-group level from accessing the `secrets` endpoint by withholding the JWT token.
- Allow users at resource-group level to create generic secrets by making a request using a different authentication mechanism. The main tenant can then validate and transform these requests before propagating them to the runtime adapter, ensuring that secret names remain consistent and critical secrets aren't unintentionally modified.

## Procedure

1. Send a POST request and enter the URL `{{apiurl1}}/v2/admin/secrets`.

- As the request body, select the `raw` radio button and enter your credentials in JSON format:

```
{  
  "name": "MY_GENERIC_SECRET",  
  "data": {  
    "some-credential": "bxktc2VjcmV0LWNyZWR1bnRpYWw=",  
    "other-credentials": "bxktc2VjcmV0LW90aGVyLWNyZWR1bnRpYWw="  
  }  
}
```

- `name`: Set the name of your generic secret.
- `data`: Enter a JSON string that represents your generic secret.

### ⓘ Note

The secret name is written without hyphens to make it simple to consume as a Unix environment variable later. It is written in capitals as is convention.

- 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 are using the resource-group level.

Key	Value
AI-Resource-Group	<Your resource group name>

- Send the request.

## Results

### ↴ Output Code

```
{  
  "message": "secret has been created",  
  "name": "MY_GENERIC_SECRET"  
}
```

## 6.6.2 Get Generic Secrets

Generic secrets can either be retrieved as a single secret, or you can list all existing secrets.

### Get a Secret Using Curl

#### Procedure

Submit a GET request to the endpoint `/v2/admin/secrets/<secret-name>`, and include the scope via the headers:

- **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 GET "$AI_API_URL/v2/admin/secrets/$SECRET_NAME" \
--header "Authorization: Bearer $TOKEN" \
--header 'AI-Resource-Group: default'
```

#### Results

The response contains the name, labels, and the creation timestamp of the requested generic secrets. No sensitive information is revealed in the response.

In the case of a tenant-wide secret, the response additionally includes a list of all resource groups associated with the tenant and the current replication status of the secret to these resource groups.

#### Output Code

```
{
  "name": "secret-1",
  "createdAt": "<timestamp>",
  "resourceGroupSecretReplicationStatus":{
    "rg-id-1" : true, # secret was replicated correctly in this namespace
    "rg-id-2" : false, # secret was not replicated or does not exist in
this namespace yet
  },
  "labels":{
    "<key>": "<value>"
  }
}
# Example response for tenant-scoped or resource group level secrets:
{
  "name": "secret-1",
  "createdAt": "<timestamp>    ",
  "labels": {
    "<key>": "<value>,
  }
```

```
}
```

## Get All Secrets Using Curl

### Procedure

Submit a GET request to the endpoint /v2/admin/secrets, and include the scope via the headers:

- **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 GET "$AI_API_URL/v2/admin/secrets" \
--header "Authorization: Bearer $TOKEN" \
--header 'AI-Resource-Group: default'
```

### Results

The response contains the name, labels, and the creation timestamp of the requested generic secrets. No sensitive information is revealed in the response.

## Get a Secret Using a Third-Party API Platform

### Procedure

Send a GET request to the endpoint {{apiurl}}/v2/admin/secrets/{{secret\_name}}.

- a. As the request body, select the **none** radio button.
- b. 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.

## Results

The response contains the name, labels, and the creation timestamp of the requested generic secrets. No sensitive information is revealed in the response.

In the case of a tenant-wide secret, the response additionally includes a list of all resource groups associated with the tenant and the current replication status of the secret to these resource groups.

### Output Code

```
{
  "name": "secret-1",
  "createdAt": "<timestamp>",
  "resourceGroupSecretReplicationStatus":{
    "rg-id-1" : true, # secret was replicated correctly in this namespace
    "rg-id-2" : false, # secret was not replicated or does not exist in
this namespace yet
  },
  "labels":{
    "<key>": "<value>",
  }
}
# Example response for tenant-scoped or resource group level secrets:
{
  "name": "secret-1",
  "createdAt": "<timestamp>",
  "labels": {
    "<key>": "<value>",
  }
}
```

## Get All Secrets Using a Third-Party API Platform

### Procedure

Send a GET request to the endpoint {{apiurl}}/v2/admin/secrets.

- a. As the request body, select the *none* radio button.
- b. 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.

## Results

The response contains the name, labels, and the creation timestamp of the requested generic secrets. No sensitive information is revealed in the response.

## 6.6.3 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": "bxkta2Vuc210aXZlWRhdGE="
  }
  "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.

## 6.6.4 Delete a Generic Secret

To get a secret name, see [Get Generic Secrets \[page 116\]](#).

### 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$AI_API_URL/v2/admin/secrets/$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 `{apiurl1}/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

### 6.6.5 Consume Generic Secrets in Executions or Deployments

Generic secrets at resource-group level can be attached to containers in executions or deployments. They can either be mounted as a volume or attached as an environment variable. The following examples illustrate how to consume a generic secret in a container by declaring it in the template. Note that only generic secrets can be attached to containers in this way. System secrets can't be consumed in a template.

#### Consume a Generic Secret as an Environment Variable

Generic secrets can be attached to containers using either `envFrom.secretRef` or `env.valueFrom.secretKeyRef`:

- Using `envFrom.secretRef`:

```
spec:  
  containers:  
    - name: my-kserve-container  
      image: centaur  
      envFrom:  
        - secretRef:  
            name: MY_GENERIC_SECRET
```

If your secret contains invalid characters, such as hyphens (-), this method results in error. In this case, map your secret to a valid variable name using `env.valueFrom.secretKeyRef`.

- Using `env.valueFrom.secretKeyRef`:

```
spec:  
  containers:  
    - name: kserve-container  
      image: centaur  
      env:  
        - name: MY_GENERIC_SECRET  
          valueFrom:  
            secretKeyRef:  
              name: my-generic-secret  
              key: some-credential
```

## Consume a Generic Secret as a Volume Mount

Generic secrets can also be mounted to containers as volumes:

```
spec:  
  containers:  
    - name: kserve-container  
      image: centaur  
      volumeMounts:  
        - name: my-generic-secret  
          mountPath: "/etc/my-generic-secret"  
          readOnly: true  
      volumes:  
        - name: my-generic-secret  
          secret:  
            secretName: my-generic-secret
```

## Additional Information

Secret names can be included as parameters in the templates and supplied via AI API configurations:

```
envFrom:  
- secretRef:  
  name: "{{inputs.parameters.secretName}}"
```

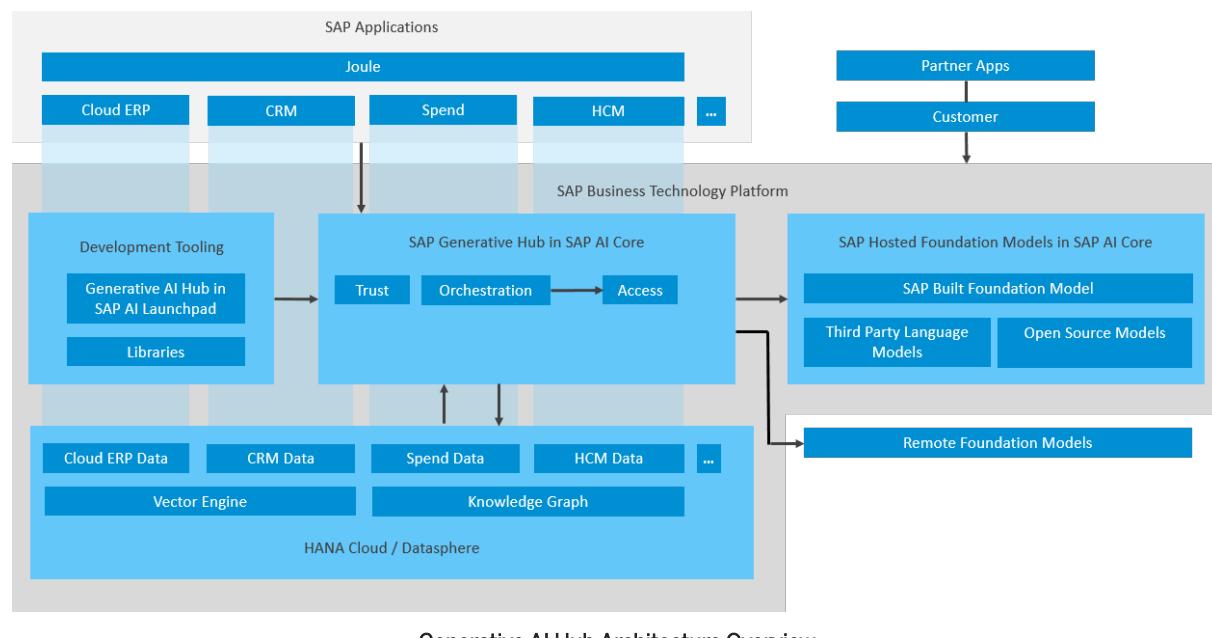
# 7 Generative AI Hub in SAP AI Core

The generative AI hub incorporates generative AI into your AI activities in SAP AI Core and SAP AI Launchpad.

Generative AI models are self-supervised, deep learning models trained on vast amounts of unlabeled data. They use AI technology and industrial-scale computational resources to learn complex patterns and semantic knowledge bases. These models excel in tasks like natural language processing (NLP). By parsing inputs, such as prompts, and predicting target words, they return contextually relevant responses in natural language. A single model can handle multiple tasks by using different input formats and output modes.

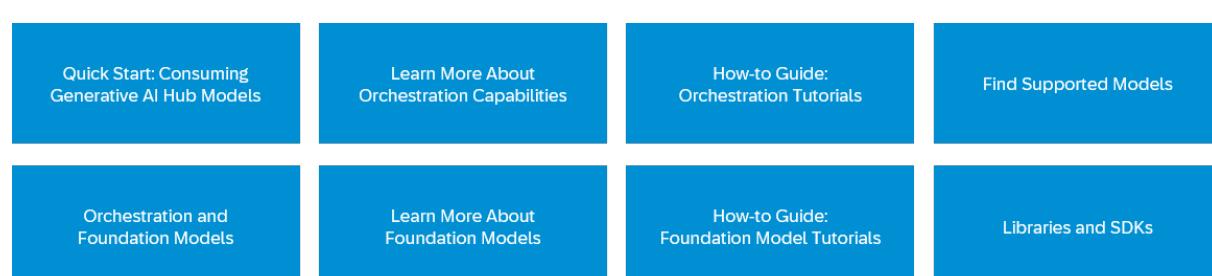
Generative AI models are general by design, but you can fine-tune them with additional embeddings. In this way, you can make them suitable for specialized or domain-specific use cases.

SAP AI Core and the generative AI hub help you to integrate LLMs and AI into new business processes in a cost-efficient manner.



## Next Steps

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



- [Quick Start \[page 126\]](#)
- [Orchestration \[page 132\]](#)
- [Orchestration Tutorials \[page 393\]](#)
- [Supported Models \[page 418\]](#)
- [Accessing Generative AI Models Through Global Scenarios \[page 131\]](#)
- [Foundation Models \[page 393\]](#)
- [Foundation Model Tutorials \[page 418\]](#)
- [Libraries and SDKs \[page 573\]](#)

## Related Information

[Monitoring and Troubleshooting \[page 598\]](#)

## 7.1 Quick Start

In this section, we'll show you how you can get started with the generative AI hub.

### Prerequisites

You have completed the steps described under [Initial Setup \[page 57\]](#).

The generative AI hub is available only with the extended service plan.

### Procedure

To get started, complete the following steps:

1. [Get an Auth Token \[page 127\]](#): This step provides instructions on how to obtain an authentication token, which is necessary for accessing SAP AI Core services securely.
2. [Get Your Orchestration Deployment URL \[page 128\]](#): This step guides you through retrieving the URL for your orchestration deployment, which is essential for model consumption.
3. [Consume Models with the Harmonized API \[page 129\]](#): This step explains how to use the Harmonized API to interact with and consume your deployed AI models without creating multiple deployments.
4. [Next Steps \[page 131\]](#): This section outlines the subsequent actions you can take after completing the initial simple model consumption, such as exploring further orchestration capabilities (grounding, content filtering, data masking)

## Related Information

[Service Plans \[page 64\]](#)

### 7.1.1 Get an Auth Token

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

#### Prerequisites

curl is likely to be installed on your operating system by default. To check, open a command prompt and enter `curl -v`. If curl isn't installed, download and install it from <https://curl.se/>.

##### Note

On macOS, you may need to install jq so that you can follow the curl commands.

1. Install brew from <https://brew.sh/>.
2. In a Terminal session, run `brew install jq` to install jq in your shell environment.

#### 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 auth token by sending the following request:

```
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...
```

## 7.1.2 Get Your 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 \[page 64\]](#) and [Upgrade a Service Plan \[page 67\]](#).
- You have completed the client authorization for your preferred user interface. For more information, see [Use a Service Key \[page 75\]](#).

### 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 \[page 421\]](#)

## 7.1.3 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.

```
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. For more information about modules available for consumption using the Harmoized API, see [Orchestration \[page 132\]](#).

### ↔ 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": "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
        }
    }
},
"orchestration_result": {
    "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
    }
}
}

```

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 \[page 573\]](#)

## 7.1.4 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 \[page 132\]](#).

Alternatively, you can deploy and consume models using direct model access. For more information, see [Foundation Models \[page 393\]](#).

## 7.2 Accessing Generative AI Models Through Global Scenarios

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.

### Related Information

[Orchestration \[page 132\]](#)

[Foundation Models \[page 393\]](#)

## 7.2.1 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, you need to 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 <code>orchestration</code> , 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 \[page 573\]](#).

### 7.2.1.1 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 October 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 \[page 259\]](#).

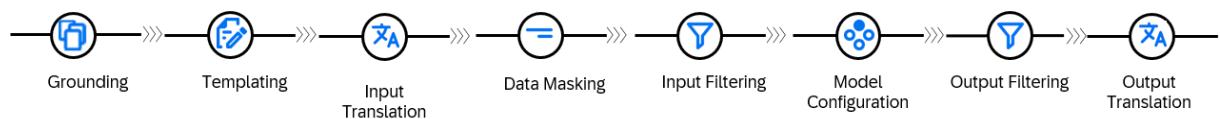
## Prerequisites

You have a running orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

## 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 \[page 265\]](#)
- [Templating \[page 221\]](#)
- [Input Translation \[page 235\]](#)
- [Data Masking \[page 237\]](#)
- [Input Filtering \[page 247\]](#)
- [Model Configuration \[page 251\]](#)
- [Output Filtering \[page 252\]](#)
- [Output Translation \[page 256\]](#)

## 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	<a href="#">Grounding [page 265]</a>
Templating (mandatory)	templating_module_config	<a href="#">Templating [page 221]</a>
Model Config (mandatory)	llm_module_config	<a href="#">Model Configuration [page 251]</a>
Content Filtering	filtering_module_config	<a href="#">Input Filtering [page 247]</a> <a href="#">Output Filtering [page 252]</a>
Data Masking	masking_module_config	<a href="#">Data Masking [page 237]</a>
Input Translation	input_translation_module_config	<a href="#">Input Translation [page 235]</a>
Output Translation	output_translation_module_config	<a href="#">Output Translation [page 256]</a>

## 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/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": {
                "threshold": 0.5
              }
            }
          ]
        }
      }
    }
  }
}'
```

```

        "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 result",
            "data": {
                "grounding_query": "grounding call",
                "grounding_result": "5 Initial Setup\nYou provision SAP AI Core from the SAP BTP cockpit in SAP Business Technology Platform. After provisioning, you will have your service key, which provides URLs and credentials for accessing the SAP AI Core instance.\nPrerequisites\nOur SAP BTP administrator has access to a global account on SAP Business Technology Platform. For more information, see Getting a Global Account .\nOur 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.\nNote\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\nThe 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 Tutorial\n. If you choose to use the\nbooster, you can skip the remaining steps to SAP AI Core Starter Tutorials [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.\nYou provision SAP AI Core from the SAP BTP cockpit in SAP Business Technology 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.\n\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.\nNote\nand 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 hub\nChoose 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\n5.2 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. You can then bind the service instance to your application, or you can create a service key to communicate\nindirectly 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```\nand 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\nindirectly 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": "Input to LLM is translated successfully.", "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.\\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 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```\nand 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\\nindirectly 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"}}, "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.\\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 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```\nand 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\\nindirectly 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"}]}]

[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 Technology 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": "Input to LLM is masked successfully.",  
 "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": "Input filter passed successfully.",
        "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": "chatmpl-BZu6RM8Z1wf74sgGu8gbtxnoSETM",
        "object": "chat.completion",
        "created": 1747896839,
        "model": "gpt-4o-2024-08-06",
        "system_fingerprint": "fp_eeld74bde0",
        "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": "0 of 1 choices failed the output filter.",
        "data": {
            "choices": [
                {
                    "index": 0,
                    "azure_content_safety": {
                        "Hate": 0,
                        "SelfHarm": 0,
                        "Sexual": 0,
                        "Violence": 0
                    }
                }
            ]
        }
    },
    "output_translation": {
        "message": "Output Translation successful",
        "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 Serviceschlüssel für die direkte Kommunikation mit der Serviceinstanz an."
                },
                "finish_reason": "stop"
            }
        ]
    }
},
"orchestration_result": {
    "id": "chatcmpl-BZuRM8Z1wf74sgGu8gbtxnoSETM",
    "object": "chat.completion",
    "created": 1747896839,
    "model": "gpt-4o-2024-08-06",
    "system_fingerprint": "fp_eeid74bde0",
    "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 Serviceschlüssel für die direkte Kommunikation mit der Serviceinstanz an."
            },
            "finish_reason": "stop"
        }
    ],
    "usage": {
        "completion_tokens": 78,
        "prompt_tokens": 722,
        "total_tokens": 800
    }
}
}

```

## 7.2.1.1 Grounding

Grounding integrates external, contextually relevant, domain-specific, or real-time data into AI processes. This data enhances the natural language processing capabilities of pretrained models, which are trained on general material.

### ⓘ Note

The grounding module configuration is consistent across orchestration versions 1 and 2.

Grounding is a service designed to handle data-related tasks, such as grounding and retrieval, using vector databases. It provides specialized data retrieval through these databases, grounding the retrieval process with

your own external and context-relevant data. Grounding combines generative AI capabilities with the ability to use real-time, precise data to improve decision-making and business operations for specific AI-driven business solutions.

Grounding converts user provided documents into vector representations which are stored as a database. The indexing pipeline preprocesses unstructured and semi structured data into chunks and embeddings. For more information, see [Pipelines API \[page 292\]](#) and [Vector API \[page 314\]](#).

The retrieval pipeline takes incoming user queries and converts them into vector representations. The query vectors are used to search the database and retrieval relevant information. For more information, see [Retrieval API \[page 332\]](#).

## Data Repositories

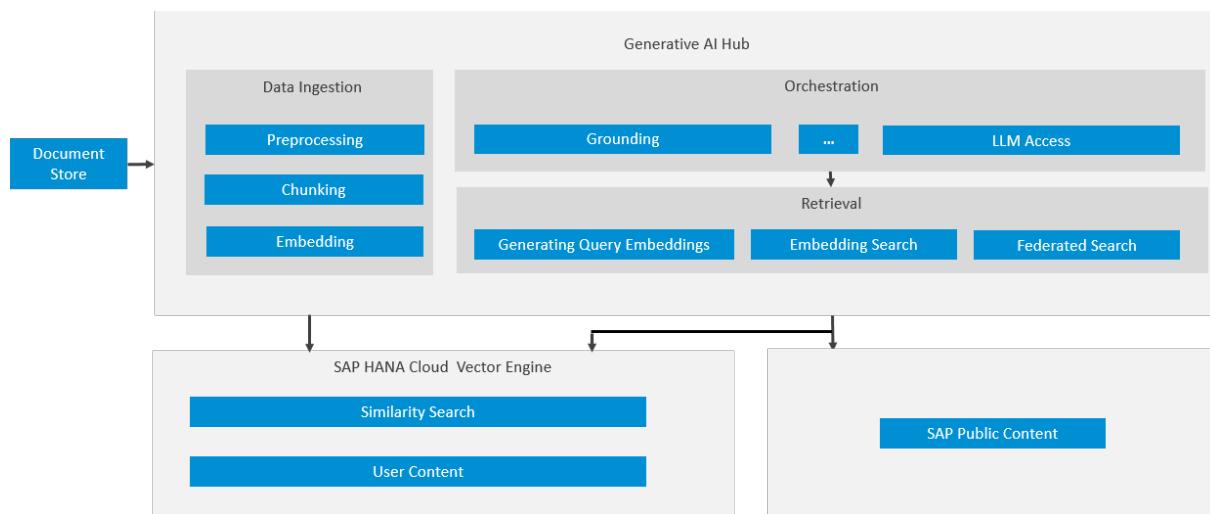
The following repositories and document types are supported:

- Microsoft SharePoint
- AWS S3
- SFTP
- SAP Build Work Zone
- SAP Document Management service

Document Repository	Document (File) Format	Content Refresh	Document Content	Maximum Number of Documents
Microsoft SharePoint	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 2000 Documents per Pipeline
AWS S3	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 2000 Documents per Pipeline
SFTP	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 2000 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 knowledge base):

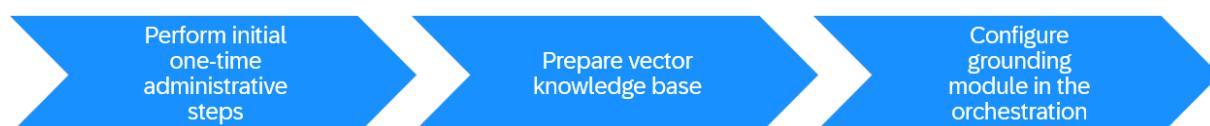
- For Option 1: Upload the documents to a supported data repository and run the data pipeline to vectorize the documents. For more information, see [Pipelines API \[page 292\]](#).
- For Option 2: Provide the chunks of document via Vector API directly. For more information, see [Vector API \[page 314\]](#).

To use grounding, 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 268]
- Prepare vector knowledge base [page 268]
- Configure grounding module in the orchestration [page 268]

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 270]
- Generic Secrets for Grounding [page 271]

## 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 \(without Metadata\) \[page 293\]](#).

## Configure Grounding Module in the Orchestration

In the orchestration pipeline, you add configuration for the grounding requests:

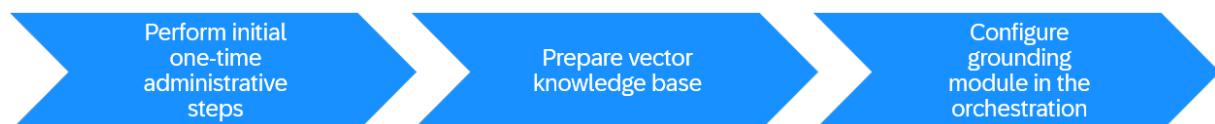
1. Create a grounding request configuration in the orchestration pipeline using the repository IDs and filters.
2. Run the orchestration pipeline and check that the response refers to the user data.

For more information, see [Using the Grounding Module \[page 337\]](#).

## 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 269]
- Prepare vector knowledge base [page 269]
- Configure grounding module in the orchestration [page 269]

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 \[page 270\]](#).

### Note

When working with the Vector API, you do not 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 \[page 315\]](#).

## 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 \[page 337\]](#).

## Next Steps

Use the retrieval API to search your data repositories.

For more information, see [Retrieval API \[page 332\]](#).

## 7.2.1.1.1 Prerequisites

### 7.2.1.1.1.1 Administrative Tasks

#### 7.2.1.1.1.1.1 Create a Resource Group for Grounding

##### Prerequisites

Resource groups represent a virtual collection of related resources within the scope of one SAP AI Core tenant. When your tenant is onboarded, a default resource group is created immediately. Further resource groups can be created, or deleted by your tenant administrator with the AI API. Tenants can map the resource groups based on the corresponding usage scenarios.

If your SAP AI Core tenant uses resource groups to isolate the scenario consumer tenant and the resource groups are subsequently deleted, the scenario consumers will be deprovisioned. SAP AI Core is not aware of the scenario consumer of the tenant. The standard XUSAA multitenancy model is followed.

For more information, see [Scope of Resources \[page 54\]](#).

##### 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 '{
    "AI-Resource-Group": "{{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 \[page 314\]](#).

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 \[page 271\]](#).

## Related Information

[Edit a Resource Group \[page 93\]](#)

[Delete a Resource Group \[page 95\]](#)

## 7.2.1.1.1.1.2 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

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 \[page 272\]](#).
- [Grounding Generic Secrets for AWS S3 \[page 276\]](#).
- [Grounding Generic Secrets for SFTP \[page 278\]](#).
- [Grounding Generic Secrets for SAP Build Work Zone \[page 280\]](#).

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 \[page 287\]](#).

## 7.2.1.1.1.1.2.1 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 \[page 270\]](#).

### 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-name>`. 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 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 <code>https://graph.microsoft.com</code>
authentication	Base64 encoded value for <code>OAuth2Password</code>
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 <code>https://login.microsoftonline.com/&lt;Microsoft Entra ID tenant identifier&gt;/oauth2/v2.0/token</code>

#### ⓘ 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
<code>name</code>	Name of the generic secret to be created
<code>description</code>	Base64 encoded value for the description of the generic secret to be created
<code>url</code>	Base64 encoded value for <code>https://sap.sharepoint.com/sites/&lt;site-name&gt;</code>
<code>authentication</code>	Base64 encoded value for <code>OAuth2ClientCredentials</code>
<code>clientId</code>	Base64 encoded value for your Microsoft SharePoint Application credentials
<code>clientSecret</code>	Base64 encoded value for your Microsoft SharePoint Application credentials
<code>tokenServiceURL</code>	Base64 encoded value for <code>https://login.microsoftonline.com/&lt;Microsoft Entra ID tenant identifier&gt;/oauth2/v2.0/token</code>

#### ⓘ 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":
    "aHR0cHM6Ly9zYXAuc2hhcmVwb2ludC5jb20vc2l0ZXMyPHNpdGUtbmFtZT4=",
    "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 \[page 292\]](#).

## Related Information

[Update a Generic Secret \[page 119\]](#)

[Delete a Generic Secret \[page 122\]](#)

## 7.2.1.1.1.1.2.2 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 \[page 270\]](#).

### Procedure

1. Send a POST request and enter the URL {{apiurl1}}/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-name>**. 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.

### 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 <b>https://s3.&lt;region&gt;.amazonaws.com</b>
authentication	Base64-encoded value for <b>NoAuthentication</b>
description	Base64-encoded value for the description of the generic secret to be created

Field	Value
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 \[page 292\]](#).

## Related Information

[Update a Generic Secret \[page 119\]](#)

[Delete a Generic Secret \[page 122\]](#)

[Making requests using the REST API ↗](#)

### 7.2.1.1.1.1.2.3 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 \[page 270\]](#).

#### 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-name>**. 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.

#### 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

Field	Value
url	Base64 encoded value in the format <b>https://&lt;hostname&gt;:&lt;port&gt;</b> where hostname and port are the SFTP server's hostname and the ports respectively
authentication	Base64 encoded value for <b>NoAuthentication</b>
description	Base64 encoded value for the description of the generic secret to be created
user	Base64 encoded value for SFTP credentials
password	Base64 encoded value for SFTP credentials
public_key	Base64 encoded value for SFTP public key, which will be used to verify the identity of the SFTP server during the connection setup
public_key_type	Base64 encoded value for SFTP public key type: <b>ssh-ed25519</b> and <b>ssh-rsa</b> .

### ⓘ 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": "Tm9BdXRoZW50aNhdGlvbg==",
    "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 \[page 292\]](#).

## Related Information

[Update a Generic Secret \[page 119\]](#)

[Delete a Generic Secret \[page 122\]](#)

## 7.2.1.1.1.1.2.4 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 have created a resource group for grounding. For more information, see [Create a Resource Group for Grounding \[page 270\]](#).

### Procedure

1. Send a POST request and enter the URL {{apiurl1}}/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 CoreChoose 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.

## 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
<code>name</code>	Name of the generic secret to be created
<code>description</code>	Base64 encoded value for the description of the generic secret to be created
<code>url</code>	Base64 encoded value for your SAP Build Work Zone URL
<code>authentication</code>	Base64 encoded value for <a href="#">OAuth2ClientCredentials</a>
<code>clientId</code>	Base64 encoded value for client ID
<code>clientSecret</code>	Base64 encoded value for client secret
<code>tokenServiceURL</code>	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": "T0F1dGgyQ2xpZW50Q3J1ZGVudGlhbHM=",
    "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": "SAP Build Work Zone"
    }
  ]
}'
```

## 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 \[page 292\]](#).

## Related Information

[Update a Generic Secret \[page 119\]](#)

[Delete a Generic Secret \[page 122\]](#)

## 7.2.1.1.1.1.2.5 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 \[page 270\]](#).

### Procedure

1. Send a POST request and enter the URL `{{apiurl1}}/v2/admin/secrets`.
  - **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.

### 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 SAP Document Management service URL
authentication	Base64 encoded value for <b>OAuth2ClientCredentials</b>
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": "T0F1dGgyQ2xpZW50Q3J1ZGVudGlhbHM=",
    "url": "< base64 encoded workzone-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": "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 \[page 292\]](#).

## Related Information

[Update a Generic Secret \[page 119\]](#)

[Delete a Generic Secret \[page 122\]](#)

### 7.2.1.1.1.1.3 Contextualization Using Metadata

#### 7.2.1.1.1.1.3.1 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 \[page 287\]](#).

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	<p>true: returns the total number of documents available on the server in the <code>documentsCount</code> field.</p> <p>false: returns the number of documents included in the current response in the <code>documentsCount</code> field.</p>

### 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 \[page 287\]](#)

### 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](#).

### 7.2.1.1.1.1.3.2 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 \[page 270\]](#).

#### Procedure

1. Send a POST request and enter the URL {{apiurl1}}/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 CoreChoose 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
<code>name</code>	Name of the generic secret to be created
<code>description</code>	Base64 encoded value for the description of the generic secret to be created
<code>url</code>	Base64 encoded value for the metadata API server URL
<code>authentication</code>	Base64 encoded value for <a href="#">OAuth2ClientCredentials</a>
<code>clientId</code>	Base64 encoded value for client ID
<code>pageSize</code> (optional)	If pagination is required, enter a Base64 encoded value of integer from 1 to 1000. <div style="border: 1px solid #e67e22; padding: 10px; margin-top: 10px;"><p><b>⚠ Restriction</b></p><p>Pagination and the page size attribute are only supported if you have a metadata server with pagination implemented. For more information, see <a href="#">Prepare your Metadata API Server [page 284]</a>.</p></div>

### With Client Secret

Field	Value
<code>clientSecret</code>	Base64 encoded value for client secret
<code>tokenServiceURL</code>	Base64 encoded value for token service URL for client secret based authentication

### With Client Certificate

Field	Value
<code>certificateType</code>	Base64 encoded value for P12 or PEM
<code>certificateContent</code>	Base64 encoded value for certificate content
<code>certificatePassword</code> (optional)	Base64 encoded value for password of the certificate if set
<code>tokenServiceURL</code>	Base64 encoded value for token service URL for certificate based authentication
<code>tokenServiceBodyParameters</code> (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": "T0F1dGgyQ2xpZW50Q3J1ZGVudGlhbHM=",
    "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": "T0F1dGgyQ2xpZW50Q3J1ZGVudGlhbHM=",
    "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"
      }
    ]
  }
}'
```

```

"data": {
    "url": "<url>",
    "description": "<description of generic secret>",
    "authentication": "T0FlIdGgyQ2xpZW50Q3JlZGVudGlhbHM=",
    "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": "T0FlIdGgyQ2xpZW50Q3JlZGVudGlhbHM=",
        "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": "T0F1dGgyQ2xpZW50Q3J1ZGVudGlhbHM=",
    "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"
    }
  ]
}'
```

## 7.2.1.1.1.2 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, preprocesses and chunks those documents, and stores their semantic embeddings in the HANA Vector Store. For more information, see [Pipelines API \[page 292\]](#).
- 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 \[page 314\]](#).

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 \[page 315\]](#).

→ 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](#).

## 7.2.1.1.1.2.1 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

Document Repository	Document (File) Format	Content Refresh	Document Content	Maximum Number of Documents
Microsoft SharePoint	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 2000 Documents per Pipeline
AWS S3	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 2000 Documents per Pipeline

Document Repository	Document (File) Format	Content Refresh	Document Content	Maximum Number of Documents
SFTP	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 2000 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.

## 7.2.1.1.1.2.1.1 Create a Document Grounding Pipeline Using the Pipelines API (without Metadata)

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 \[page 270\]](#).
- You have created a generic secret. For more information, see [Generic Secrets for Grounding \[page 271\]](#).
- You're using a supported data source and document type. For more information, see [Pipelines API \[page 292\]](#).

## Context

### → 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

1. Send a POST request to the endpoint \$AI\_API\_URL/v2/lm/document-grounding/pipelines.  
**The following examples show requests for each data source:**

### Microsoft SharePoint

The following shows an example of a SharePoint site to be indexed.

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
<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.

## AWS S3

The following shows an example of an AWS S3 repository to be indexed:

Populate the sample code with the following values:

Value	
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{{access_token}}</code>	<code>{{access_token}}</code>
<code>&lt;generic secret name&gt;</code>	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" ]
    }
  }
}'
```

## SFTP

The following shows an example of an SFTP repository to be indexed:

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
<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"]
    }
  }
}'
```

## SAP Build Work Zone

The following shows an example of a SAP Build Work Zone site to be indexed.

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
<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": "SAP Build Work Zone",
  "metadata": {
    "destination": "<generic secret name>"
  }
}'
```

## SAP Document Management Service

The following shows an example of a SAP Document Management service site to be indexed.

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
<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>"
  }
}'
```

## 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 \[page 332\]](#) or [Using the Grounding Module \[page 337\]](#).

## 7.2.1.1.1.2.1.2 Create a Document Grounding Pipeline Using the Pipelines API (with Metadata)

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 \[page 270\]](#).
- You have created a generic secret. For more information, see [Generic Secrets for Grounding \[page 271\]](#).
- You're using a supported data source and document type. For more information, see [Pipelines API \[page 292\]](#).

### Context

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 \(without Metadata\) \[page 293\]](#).

→ 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

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines`.  
The following examples show requests for each data source:

### Microsoft SharePoint

The following shows an example of a SharePoint site to be indexed with metadata.

Populate the sample code with the following values:

Value	
<code>{{resource_group}}</code>	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
<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>"
  }
}'
```

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.

To make your pipeline searchable, add dataRepositoryMetadata to the metadata field. For example:

```
...
  "metadata": {
    "dataRepositoryMetadata": [
      {
        "key": "example",
        "value": [
          "demo"
        ]
      }
    ]
  }
...
```

For more information, see [Search a Pipeline \[page 313\]](#).

## AWS S3

The following shows an example of an AWS S3 repository to be indexed with metadata:

Populate the sample code with the following values:

	Value
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{access_token}</code>	Your access token for SAP AI Core
<code>&lt;generic secret name&gt; (configuration)</code>	The name of the generic secret created for AWS S3
<code>&lt;generic secret name&gt; (metadata)</code>	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>"
  }
}'
```

The `metadata` attribute is optional. It accepts the destination name, which is used to connect to the AWS S3 metadata server to retrieve metadata for document indexing.

To make your pipeline searchable, add `dataRepositoryMetadata` to the `metadata` field. For example:

```
...
  "metadata": {
    "dataRepositoryMetadata": [
      {
        "key": "example",
        "value": [
          "demo"
        ]
      }
    ]
  }
...
...
```

For more information, see [Search a Pipeline \[page 313\]](#).

## SFTP

The following shows an example of an SFTP repository to be indexed with metadata:

Populate the sample code with the following values:

	Value
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{{access_token}}</code>	Your access token for SAP AI Core
<code>&lt;generic secret name&gt; (configuration)</code>	The name of the generic secret created for SFTP
<code>&lt;generic secret name&gt; (metadata)</code>	The name of the generic secret created for the SFTP Meta-Data 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.

To make your pipeline searchable, add `dataRepositoryMetadata` to the `metadata` field. For example:

```
...
  "metadata": {
    "dataRepositoryMetadata": [
      {
        "key": "example",
        "value": [
          "demo"
        ]
      }
    ]
  }
```

```
        }
    ]
...
}
```

For more information, see [Search a Pipeline \[page 313\]](#).

## 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 \[page 332\]](#) or [Using the Grounding Module \[page 337\]](#).

### 7.2.1.1.1.2.1.3 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/v2/1m/document-grounding/pipelines`

You can use the endpoint `$AI_API_URL/v2/1m/document-grounding/pipelines` to retrieve all document grounding pipelines.

Populate the code snippet with the following:

- `{resource_group}`: the AI resource group ID assigned to your account.
- `AI_API_URL`: the base URL of your SAP AI Core environment.
- `{access_token}`: Your access token for SAP AI Core

### 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.

The available statuses for pipelines are the following:

- 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.

For example:

```
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}}'
```

## 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 (for 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/v2/1m/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/v2/1m/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.

The available statuses for pipelines are the following:

- 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.

The available statuses for pipelines are the following:

- 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.

## 7.2.1.1.1.2.1.4 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
<code>\$top</code>	Integer	Limits the number of results returned. For example, <code>\$top=10</code> returns 10 results.
<code>\$skip</code>	Integer	Skips a specific number of results. For example, <code>\$skip=20</code> skips the first 20.
<code>\$count</code>	Boolean	If <code>true</code> , includes the total count in the response. If <code>false</code> , includes the total number of records in the response.

## ⌚ Example

This code returns a JSON array of pipeline execution objects.

```
curl --request GET  
  --url "$AI_API_URL/v2/1m/document-grounding/pipelines/{{pipelineId}}/  
executions?$top=10&$skip=20&$count=true"  
  --header "AI-Resource-Group: {{resource_group}}"  
  --header 'Authorization: Bearer {{access_token}}'
```

The available statuses for executions are the following:

- 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/v2/1m/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/v2/1m/document-grounding/pipelines/{{pipelineId}}/  
executions/{{executionId}}" \  
  --header "AI-Resource-Group: {{resource_group}}"  
  --header 'Authorization: Bearer {{access_token}}'
```

The available statuses for executions are the following:

- NEW
- UNKNOWN
- INPROGRESS

- FINISHED
- FINISHEDWITHERRORS
- TIMEOUT

## 7.2.1.1.1.2.1.5 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}}'
```

The available statuses for documents are the following:

- TO\_BE\_PROCESSED
- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED\_TO\_BE\_RETRYED
- 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}}'
```

The available statuses for documents are the following:

- TO\_BE\_PROCESSED
- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED\_TO\_BE\_RETRYED
- 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}}'
```

The available statuses for documents are the following:

- TO\_BE\_PROCESSED
- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED\_TO\_BE\_RETRYED
- 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}}'
```

The available statuses for documents are the following:

- TO\_BE\_PROCESSED

- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED\_TO\_BE\_RETRYED
- 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}}'
```

The available statuses for documents are the following:

- TO\_BE\_PROCESSED
- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED\_TO\_BE\_RETRYED
- TO\_BE\_SCHEDULED

## 7.2.1.1.1.2.1.6 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

## ↔ 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}}"
}'
```

## Results

The response includes a response code. Code 202 is a successful response.

### 7.2.1.1.1.2.1.7 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 \(with Metadata\) \[page 298\]](#).

#### 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": "value"
    }
  ]
}'
```

```

        "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": "<sname>"
                }
            }
        }
    ]
}

```

## 7.2.1.1.1.2.2 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 chunk is a small segment of a document's text, used in the Vector API to improve the accuracy and efficiency of semantic search.

## 7.2.1.1.1.2.2.1 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 \[page 270\]](#).

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 \[page 316\]](#).
2. Insert documents and chunks into the collection. For more information see [Create a Document \[page 323\]](#).

### 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 \[page 332\]](#) or [Using the Grounding Module \[page 337\]](#).

Alternatively, you can search specific collections using vector search. For more information, see [Vector Search \[page 330\]](#)

## 7.2.1.1.1.2.2.1.1 Collections

### 7.2.1.1.1.2.2.1.1.1 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 \[page 270\]](#)

#### 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:

	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
<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

#### ① Note

The `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 '{
  "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 \[page 320\]](#).

### 7.2.1.1.1.2.2.1.1.2

### 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.

### 7.2.1.1.1.2.2.1.1.3 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.

### 7.2.1.1.1.2.2.1.1.4

### 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
<code>{{resource_group}}</code>	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.

### 7.2.1.1.1.2.2.1.1.5 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 the status of your collection creation.

## Next Steps

After the collection has been created successfully, you can add documents to your collection. For more information, see [Create a Document \[page 323\]](#).

### 7.2.1.1.1.2.2.1.1.6 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.

### 7.2.1.1.1.2.2.1.1.7 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/v2/1m/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/v2/1m/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 the status of your collection deletion.

## 7.2.1.1.1.2.2.1.2 Documents

### 7.2.1.1.1.2.2.1.2.1 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 \[page 314\]](#) and [Create a Collection \[page 316\]](#).

You know the ID of the collection that you want to add documents to. For more information, see [Get a Collection \[page 319\]](#).

#### Procedure

- 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	
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{{access_token}}</code>	Your access token for SAP AI Core
<code>Content-Type</code>	<code>application/json</code>
<code>{collectionId}</code>	The ID of the collection
<code>metadata &lt;key&gt; &lt;value&gt; pair</code>	Your choice of metadata key value pair
<code>chunks</code>	Text content in chunks
<code>metadata &lt;key&gt; &lt;value&gt; pair (chunks)</code>	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 \[page 330\]](#).

To search for relevant chunks across all collections based on a user query, use the Retrieval API. For more information, see [Retrieval API \[page 332\]](#).

To use your documents as part of an orchestration workflow, see [Using the Grounding Module \[page 337\]](#).

## 7.2.1.1.1.2.2.1.2.2 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
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{{access_token}}</code>	Your access token for SAP AI Core
<code>Content-Type</code>	<code>application/json</code>
<code>{{collectionId}}</code>	The ID of the collection
<code>{{documentId}}</code>	The ID of the document
metadata <code>&lt;key&gt; &lt;value&gt;</code> pair	Your choice of metadata key value pair
chunks	Text content in chunks
metadata <code>&lt;key&gt; &lt;value&gt;</code> 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": "example",
          "value": "value"
        }
      ]
    }
  ]
}'
```

```
        "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 response 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

## 7.2.1.1.1.2.2.1.2.3 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.

## 7.2.1.1.1.2.2.1.2.4 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
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{{access_token}}</code>	Your access token for SAP AI Core
<code>{{collectionId}}</code>	The ID of the collection
<code>{{documentId}}</code>	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.

## 7.2.1.1.1.2.2.1.2.5 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
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{{access_token}}</code>	Your access token for SAP AI Core
<code>{{collectionId}}</code>	The ID of the collection
<code>{{documentId}}</code>	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

## 7.2.1.1.1.2.2.2 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.

### Procedure

Send a POST request to the endpoint: \$AI\_API\_URL/v2/1m/document-grounding/vector/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
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

#### ↔ Sample Code

```
curl --request POST \
--url $AI_API_URL/v2/1m/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": []
    }
  ]
}'
```

```
}
```

## Results

The response is returned in JSON format, and includes relevant chunks based on similarity search..

## 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
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

Attribute	Type	Required	Description	Constraints
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

### 7.2.1.1.1.2.3 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 \[page 314\]](#).

#### 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](#).

### 7.2.1.1.1.2.3.1 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/v2/1m/document-grounding/retrieval/dataRepositories`.

Populate the sample code with the following values:

	Value
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{{access_token}}</code>	Your access token for SAP AI Core

### ↔ Sample Code

```
curl --request GET \
--url $AI_API_URL/v2/1m/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
<code>\$top</code>	Integer	Limits the number of results returned. For example, <code>\$top=10</code> returns 10 results.
<code>\$skip</code>	Integer	Skips a specific number of results. For example, <code>\$skip=20</code> skips the first 20.
<code>\$count</code>	Boolean	If <code>true</code> , includes the total count in the response. If <code>false</code> , 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}}'
```

### 7.2.1.1.1.2.3.2 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
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{{access_token}}</code>	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.

## 7.2.1.1.1.2.3.3 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
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{{access_token}}</code>	Your access token for SAP AI Core
<code>&lt;user query&gt;</code>	The query that you want to be answered from the vector database
<code>dataRepositories</code>	'*' searches through all collections. Alternatively, enter a collection ID to retrieve a chunk from that collection.
<code>dataRepositoryType</code>	Supported repository types are <code>vector</code> and <code>help.sap.com</code> . Data from <code>help.sap.com</code> 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.

## 7.2.1.1.2 Using the Grounding Module

### Prerequisites

You have a running orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

### 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.
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 \[page 573\]](#)

## 7.2.1.1.2.1 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 orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

You have onboarded a repository. For more information, see [Generic Secrets for Grounding \[page 271\]](#).

### 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": "geminii-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_template": {
    "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>"
}

```

## 7.2.1.1.2.2 Retrieval Using help.sap.com

### Prerequisites

You have a running orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

You have onboarded a repository. For more information, see [Generic Secrets for Grounding \[page 271\]](#).

### 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": [
          "*"
        ],
        "dataRepositoryType": "help.sap.com",
        "documentMetadata": []
      }
    ],
    "documentMetadata": [],
    "chunkMetadata": []
}'
```

The response includes details of metadata parameters on the following levels: `dataRepository`, `document` and `chunk`.

- 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-raw '{
    "orchestration_config": {
        "module_configurations": [
            "grounding_module_config": {
                "type": "document_grounding_service",
                "config": {
                    "filters": [
                        {
                            "id": "filter1",
                            "data_repositories": [
                                "*"
                            ],
                            "search_config": {},
                            "data_repository_type": "help.sap.com",
                            "document_metadata": []
                        }
                    ]
                }
            },
            "input_params": [
                "groundingRequest",
                "metadata"
            ],
            "output_param": "groundingOutput"
        }
    },
    "llm_module_config": {
        "model_name": "<modelName>",
        "model_params": {},
        "model_version": "<modelVersion>"
    },
    "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 }}\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?"
        }
    }
}

```

### 7.2.1.1.2 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 \[page 423\]](#).

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"data:image/jpeg;base64,{base64_image}"
}
```

or

```
"image_url": {  
    "url": "https://some.jpg"  
}
```

## 7.2.1.1.2.1 Few-Shot Learning

### Prerequisites

You have a running orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

### 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": "assistant",  
                        "content": "Tech"  
                    }  
                ]  
            }  
        }  
    }  
}
```

```

        "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 \[page 573\]](#)

### 7.2.1.1.2.2 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 orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

## 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."
          }
        ],
        "ilm_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-9kXqisJKnuNv1B4eXTUzqZEJSmdC",
      "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": 100
      }
    }
  }
}
```

```

        "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 \[page 573\]](#)

### 7.2.1.1.2.3 Minimal Call

## Prerequisites

You have a running orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

## 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",
      "result": "Orchestration is Working! in German"
    }
  }
}'
```

```

    "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 \[page 573\]](#)

### 7.2.1.1.2.4 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.

## 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-BKP9iN0ustKi2OA7efssSi5yw31EO4",
    "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_i13KDaSC5zm6naTOnYv5VSZT",
                        "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
          }
        }
      }
    ]
  },
  "ilm_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-BKPKM10RalAlw2drake8GUfSe60sZ",
      "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-BKPKM10Ra1Alw2drape8GUfSe60sz",
    "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
    }
}
```

### 7.2.1.1.3 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.

#### 7.2.1.1.3.1 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!"
  }
}
}

```

## 7.2.1.1.4 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 \[page 245\]](#).

#### ⚠ 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.

### 7.2.1.1.4.1 Enhancing Model Consumption with Data Masking

#### Prerequisites

You have a running orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

#### 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
patrick.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 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.

## 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\npatrick.morgan@hotmail.com\nHighlights\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\nEducation\nMaster of Science: Finance - 2014\nHarvard University, Boston\nBachelor of Science: Finance - 2011\nHarvard University, Boston\n\nCertifications\nCertified Management Accountant\nSummary\nSkilled 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\nExperience\nFinance Manager - 09/2016 to 05/2018\nM&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\nFinance Manager - 09/2013 to 05/2016\nAgo 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": "Input to LLM is masked successfully.",
      "data": {
        "masked_template": [
          {
            "role": "user",
            "content": "Summarize the following CV in 10 sentences:\nMASKED_PERSON\nMASKED_PHONE_NUMBER\nMASKED_EMAIL\nHighlights\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\nEducation\nMaster of Science: Finance - 2014\nHarvard University, Boston\nBachelor of Science: Finance - 2011\nHarvard University, Boston\n\nCertifications\nCertified Management Accountant\nSummary\nSkilled 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\nExperience\nFinance Manager - 09/2016 to 05/2018\nMASKED_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 Certifications \n Certified Management Accountant \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 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": "Input to LLM is masked successfully.",
            "data": [
                {
                    "masked_template": [
                        {
                            "role": "user",
                            "content": "Summarize the 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 Certifications \n Certified Management Accountant \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."
    ]
}
},
"11m": {
  "id": "chatcmpl-A5ve0Hbkabo04FugU7ysHeWeH7ZZC",
  "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-A5ve0Hbkabo04FugU7ysHeWeH7ZZC",
  "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 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
    }
}

```

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 \[page 573\]](#)

## 7.2.1.5 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 \[page 248\]](#)

### 7.2.1.1.5.1 Enhancing Model Consumption with Input Filtering

## Prerequisites

You have a running orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

## 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": "Content filtered due to Safety violations. Please modify the prompt and try again.",
  "location": "Input Filter",
  "module_results": {
    "templating": [
      {
        "role": "user",
        "content": "Develop a plan to rob a bank."
      }
    ],
    "input_filtering": {
      "message": "Content filtered due to Safety violations. Please modify the prompt and try again.",
      "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 [page 573]

### 7.2.1.1.6 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 \[page 393\]](#) 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 \[page 379\]](#).  
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 600, 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  
}
```

## 7.2.1.1.7 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 \[page 253\]](#)

### 7.2.1.1.7.1 Enhancing Model Consumption with Output Filtering

#### Prerequisites

You have a running orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

#### 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": "Input filter passed successfully.",
      "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": "1 of 1 choices failed the output filter.",
      "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 \[page 573\]](#)

### 7.2.1.1.8 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.

## 7.2.1.1.8.1 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!"
    }
}
```

## 7.2.1.2 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 orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

### 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	<a href="#">Grounding [page 265]</a>
Templating (mandatory)	prompt_templating	<a href="#">Templating [page 344]</a>
Content Filtering	filtering	<a href="#">Content Filtering [page 371]</a>
Data Masking	masking	<a href="#">Data Masking [page 364]</a>
Translation	translation	<a href="#">Translation [page 360]</a>

## 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/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_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": {
    "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  
kontakteien Sie mich mit unter Jane.Janeson@gmx.net."  
    }  
}'
```

## ↔ Output Code

```
{  
  "request_id": "527798de-d43b-93ee-b4bd-6111945ae074",  
  "intermediate_results": {  
    "grounding": {  
      "message": "grounding result",  
      "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": "Input to LLM is translated successfully.",  
      "data": {  
        "translated_template": "..."  
      }  
    },  
    "input_masking": {  
      "message": "Input to LLM is masked successfully.",  
      "data": {  
        "masked_template": "..."  
      }  
    },  
    "input_filtering": {  
      "message": "Input Filter passed successfully.",  
      "data": {  
        "azure_content_safety": {  
          "Hate": 0,  
          "SelfHarm": 0,  
          "Sexual": 0,  
          "Violence": 0  
        }  
      }  
    },  
    "output_filtering": {  
      "message": "0 of 1 choices failed the output filter.",  
      "data": {  
        "choices": [  
        ]  
      }  
    }  
  }  
}
```

```

        "index": 0,
        "azure_content_safety": {
            "Hate": 0,
            "SelfHarm": 0,
            "Sexual": 0,
            "Violence": 0
        }
    }
}
},
"output_translation": {
    "message": "Output Translation successful",
    "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-BtyoOco09vPl1IbBZVYtIOkUJaGJ",
    "object": "chat.completion",
    "created": 1752681500,
    "model": "gpt-4o-2024-08-06",
    "system_fingerprint": "fp_eeld74bde0",
    "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[Your Support Team]"
            }
        }
    ]
}
}

```

```

        },
        "finish_reason": "stop"
    }
],
"usage": {
    "completion_tokens": 196,
    "prompt_tokens": 1621,
    "total_tokens": 1817
}
},
"final_result": {
    "id": "chatcmpl-BtyoOco09vPl1IbBZVyyTIOkUJaGJ",
    "object": "chat.completion",
    "created": 1752681500,
    "model": "gpt-4o-2024-08-06",
    "system_fingerprint": "fp_eeld74bde0",
    "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
}
}
}

```

## 7.2.1.2.1 Grounding

Grounding integrates external, contextually relevant, domain-specific, or real-time data into AI processes. This data enhances the natural language processing capabilities of pretrained models, which are trained on general material.

### ⓘ Note

The grounding module configuration is consistent across orchestration versions 1 and 2.

Grounding is a service designed to handle data-related tasks, such as grounding and retrieval, using vector databases. It provides specialized data retrieval through these databases, grounding the retrieval process with your own external and context-relevant data. Grounding combines generative AI capabilities with the ability to use real-time, precise data to improve decision-making and business operations for specific AI-driven business solutions.

Grounding converts user provided documents into vector representations which are stored as a database. The indexing pipeline preprocesses unstructured and semi structured data into chunks and embeddings. For more information, see [Pipelines API \[page 292\]](#) and [Vector API \[page 314\]](#).

The retrieval pipeline takes incoming user queries and converts them into vector representations. The query vectors are used to search the database and retrieval relevant information. For more information, see [Retrieval API \[page 332\]](#).

## Data Repositories

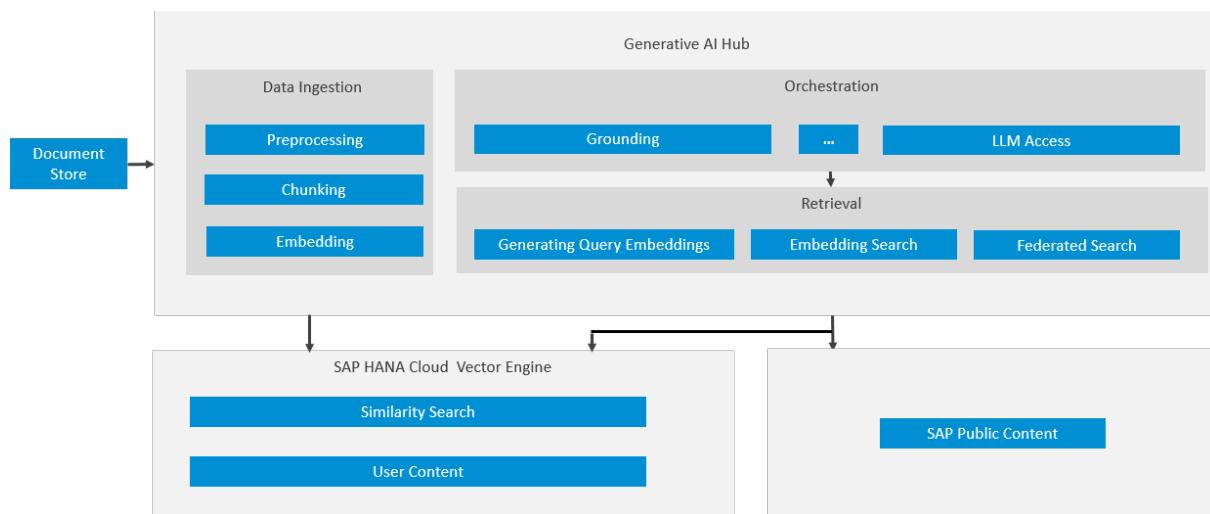
The following repositories and document types are supported:

- Microsoft SharePoint
- AWS S3
- SFTP
- SAP Build Work Zone
- SAP Document Management service

Document Repository	Document (File) Format	Content Refresh	Document Content	Maximum Number of Documents
Microsoft SharePoint	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 2000 Documents per Pipeline
AWS S3	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 2000 Documents per Pipeline
SFTP	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 2000 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 knowledge base):

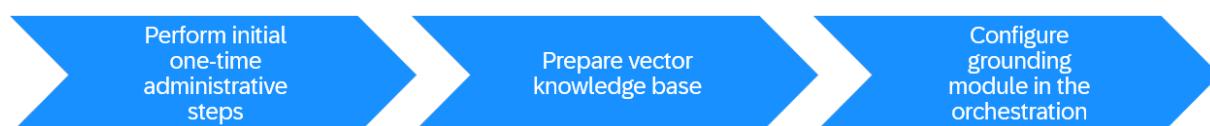
- For Option 1: Upload the documents to a supported data repository and run the data pipeline to vectorize the documents. For more information, see [Pipelines API \[page 292\]](#).
- For Option 2: Provide the chunks of document via Vector API directly. For more information, see [Vector API \[page 314\]](#).

To use grounding, 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 268]
- Prepare vector knowledge base [page 268]
- Configure grounding module in the orchestration [page 268]

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 270]
- Generic Secrets for Grounding [page 271]

## 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 \(without Metadata\) \[page 293\]](#).

## Configure Grounding Module in the Orchestration

In the orchestration pipeline, you add configuration for the grounding requests:

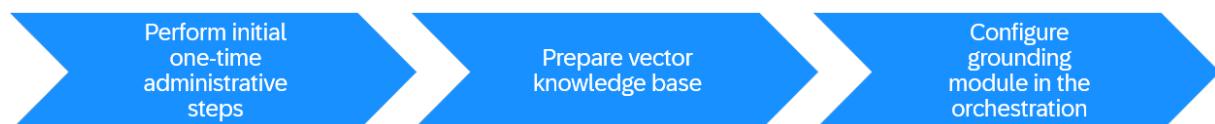
1. Create a grounding request configuration in the orchestration pipeline using the repository IDs and filters.
2. Run the orchestration pipeline and check that the response refers to the user data.

For more information, see [Using the Grounding Module \[page 337\]](#).

## 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 269]
- Prepare vector knowledge base [page 269]
- Configure grounding module in the orchestration [page 269]

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 \[page 270\]](#).

### Note

When working with the Vector API, you do not 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 \[page 315\]](#).

## 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 \[page 337\]](#).

## Next Steps

Use the retrieval API to search your data repositories.

For more information, see [Retrieval API \[page 332\]](#).

## 7.2.1.2.1.1 Prerequisites

### 7.2.1.2.1.1.1 Administrative Tasks

#### 7.2.1.2.1.1.1.1 Create a Resource Group for Grounding

##### Prerequisites

Resource groups represent a virtual collection of related resources within the scope of one SAP AI Core tenant. When your tenant is onboarded, a default resource group is created immediately. Further resource groups can be created, or deleted by your tenant administrator with the AI API. Tenants can map the resource groups based on the corresponding usage scenarios.

If your SAP AI Core tenant uses resource groups to isolate the scenario consumer tenant and the resource groups are subsequently deleted, the scenario consumers will be deprovisioned. SAP AI Core is not aware of the scenario consumer of the tenant. The standard XUSAA multitenancy model is followed.

For more information, see [Scope of Resources \[page 54\]](#).

##### 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 '{
    "AI-Resource-Group": "{{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 \[page 314\]](#).

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 \[page 271\]](#).

## Related Information

[Edit a Resource Group \[page 93\]](#)

[Delete a Resource Group \[page 95\]](#)

### 7.2.1.2.1.1.2 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

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 \[page 272\]](#).
- [Grounding Generic Secrets for AWS S3 \[page 276\]](#).
- [Grounding Generic Secrets for SFTP \[page 278\]](#).
- [Grounding Generic Secrets for SAP Build Work Zone \[page 280\]](#).

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 \[page 287\]](#).

## 7.2.1.2.1.1.2.1 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 \[page 270\]](#).

### 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-name>`. 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 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 <code>https://graph.microsoft.com</code>
authentication	Base64 encoded value for <code>OAuth2Password</code>
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 <code>https://login.microsoftonline.com/&lt;Microsoft Entra ID tenant identifier&gt;/oauth2/v2.0/token</code>

#### ⓘ 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
<code>name</code>	Name of the generic secret to be created
<code>description</code>	Base64 encoded value for the description of the generic secret to be created
<code>url</code>	Base64 encoded value for <code>https://sap.sharepoint.com/sites/&lt;site-name&gt;</code>
<code>authentication</code>	Base64 encoded value for <code>OAuth2ClientCredentials</code>
<code>clientId</code>	Base64 encoded value for your Microsoft SharePoint Application credentials
<code>clientSecret</code>	Base64 encoded value for your Microsoft SharePoint Application credentials
<code>tokenServiceURL</code>	Base64 encoded value for <code>https://login.microsoftonline.com/&lt;Microsoft Entra ID tenant identifier&gt;/oauth2/v2.0/token</code>

#### ⓘ 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":
    "aHR0cHM6Ly9zYXAuc2hhcmVwb2ludC5jb20vc2l0ZXMyPHNpdGUtbmFtZT4=",
    "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 \[page 292\]](#).

## Related Information

[Update a Generic Secret \[page 119\]](#)

[Delete a Generic Secret \[page 122\]](#)

## 7.2.1.2.1.1.2.2 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 \[page 270\]](#).

### Procedure

1. Send a POST request and enter the URL {{apiurl1}}/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-name>**. 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.

### 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 <b>https://s3.&lt;region&gt;.amazonaws.com</b>
authentication	Base64-encoded value for <b>NoAuthentication</b>
description	Base64-encoded value for the description of the generic secret to be created

Field	Value
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 \[page 292\]](#).

## Related Information

[Update a Generic Secret \[page 119\]](#)

[Delete a Generic Secret \[page 122\]](#)

[Making requests using the REST API ↗](#)

### 7.2.1.2.1.1.1.2.3 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 \[page 270\]](#).

#### 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-name>**. 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.

#### 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

Field	Value
url	Base64 encoded value in the format <b>https://&lt;hostname&gt;:&lt;port&gt;</b> where hostname and port are the SFTP server's hostname and the ports respectively
authentication	Base64 encoded value for <b>NoAuthentication</b>
description	Base64 encoded value for the description of the generic secret to be created
user	Base64 encoded value for SFTP credentials
password	Base64 encoded value for SFTP credentials
public_key	Base64 encoded value for SFTP public key, which will be used to verify the identity of the SFTP server during the connection setup
public_key_type	Base64 encoded value for SFTP public key type: <b>ssh-ed25519</b> and <b>ssh-rsa</b> .

### ⓘ 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>",
    "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 \[page 292\]](#).

## Related Information

[Update a Generic Secret \[page 119\]](#)

[Delete a Generic Secret \[page 122\]](#)

### 7.2.1.2.1.1.1.2.4 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 have created a resource group for grounding. For more information, see [Create a Resource Group for Grounding \[page 270\]](#).

## Procedure

1. Send a POST request and enter the URL {{apiurl1}}/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.

## 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
<code>name</code>	Name of the generic secret to be created
<code>description</code>	Base64 encoded value for the description of the generic secret to be created
<code>url</code>	Base64 encoded value for your SAP Build Work Zone URL
<code>authentication</code>	Base64 encoded value for <a href="#">OAuth2ClientCredentials</a>
<code>clientId</code>	Base64 encoded value for client ID
<code>clientSecret</code>	Base64 encoded value for client secret
<code>tokenServiceURL</code>	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": "T0F1dGgyQ2xpZW50Q3J1ZGVudGlhbHM=",
    "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": "SAP Build Work Zone"
    }
  ]
}'
```

## 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 \[page 292\]](#).

## Related Information

[Update a Generic Secret \[page 119\]](#)

[Delete a Generic Secret \[page 122\]](#)

### 7.2.1.2.1.1.1.2.5 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 \[page 270\]](#).

## 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 CoreChoose 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.

## 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
<code>name</code>	Name of the generic secret to be created
<code>description</code>	Base64 encoded value for the description of the generic secret to be created
<code>url</code>	Base64 encoded value for your SAP Document Management service URL
<code>authentication</code>	Base64 encoded value for <a href="#">OAuth2ClientCredentials</a>
<code>clientId</code>	Base64 encoded value for client ID
<code>clientSecret</code>	Base64 encoded value for client secret
<code>tokenServiceURL</code>	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": "T0F1dGgyQ2xpZW50Q3J1ZGVudGlhbHM=",
    "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": "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 \[page 292\]](#).

## Related Information

[Update a Generic Secret \[page 119\]](#)

[Delete a Generic Secret \[page 122\]](#)

### 7.2.1.2.1.1.1.3 Contextualization Using Metadata

#### 7.2.1.2.1.1.1.3.1 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 \[page 287\]](#).

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	<p>true: returns the total number of documents available on the server in the documentsCount field.</p> <p>false: returns the number of documents included in the current response in the documentsCount field.</p>

### 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 \[page 287\]](#)

#### 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"], "matchMode": "ANY" },
            { "key": "department", "value": ["Engineering"], "matchMode": "ALL" }
          ],
          "downloadLocation": "/path/to/download/doc3",
          "lastUpdatedTimestamp": "2024-02-26T12:00:00Z"
        }
      ]
    }
  ]
}
```

```
        "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](#).

### 7.2.1.2.1.1.3.2 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 \[page 270\]](#).

#### Procedure

1. Send a POST request and enter the URL {{apiurl1}}/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 CoreChoose 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
<code>name</code>	Name of the generic secret to be created
<code>description</code>	Base64 encoded value for the description of the generic secret to be created
<code>url</code>	Base64 encoded value for the metadata API server URL
<code>authentication</code>	Base64 encoded value for <a href="#">OAuth2ClientCredentials</a>
<code>clientId</code>	Base64 encoded value for client ID
<code>pageSize</code> (optional)	If pagination is required, enter a Base64 encoded value of integer from 1 to 1000. <div style="border: 1px solid #e67e22; padding: 10px; margin-top: 10px;"><p><b>⚠ Restriction</b></p><p>Pagination and the page size attribute are only supported if you have a metadata server with pagination implemented. For more information, see <a href="#">Prepare your Metadata API Server [page 284]</a>.</p></div>

### With Client Secret

Field	Value
<code>clientSecret</code>	Base64 encoded value for client secret
<code>tokenServiceURL</code>	Base64 encoded value for token service URL for client secret based authentication

### With Client Certificate

Field	Value
<code>certificateType</code>	Base64 encoded value for P12 or PEM
<code>certificateContent</code>	Base64 encoded value for certificate content
<code>certificatePassword</code> (optional)	Base64 encoded value for password of the certificate if set
<code>tokenServiceURL</code>	Base64 encoded value for token service URL for certificate based authentication
<code>tokenServiceBodyParameters</code> (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": "T0F1dGgyQ2xpZW50Q3J1ZGVudGlhbHM=",
    "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": "T0F1dGgyQ2xpZW50Q3J1ZGVudGlhbHM=",
    "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"
      }
    ]
  }
}'
```

```

    "data": {
      "url": "<url>",
      "description": "<description of generic secret>",
      "authentication": "T0FlIdGgyQ2xpZW50Q3JlZGVudGlhbHM=",
      "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": "T0FlIdGgyQ2xpZW50Q3JlZGVudGlhbHM=",
    "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": "T0F1dGgyQ2xpZW50Q3J1ZGVudGlhbHM=",
    "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"
    }
  ]
}'
```

## 7.2.1.2.1.1.2 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, preprocesses and chunks those documents, and stores their semantic embeddings in the HANA Vector Store. For more information, see [Pipelines API \[page 292\]](#).
- 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 \[page 314\]](#).

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 \[page 315\]](#).

→ 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](#).

## 7.2.1.2.1.2.1 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

Document Repository	Document (File) Format	Content Refresh	Document Content	Maximum Number of Documents
Microsoft SharePoint	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 2000 Documents per Pipeline
AWS S3	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 2000 Documents per Pipeline

Document Repository	Document (File) Format	Content Refresh	Document Content	Maximum Number of Documents
SFTP	PDF, HTML, TXT, JPEG, JPG, DOCX, PNG, TIFF, PPT	Daily	Plain Text PPT only: Images, Plain Text, Table	Maximum of 2000 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.

## 7.2.1.2.1.1.2.1.1 Create a Document Grounding Pipeline Using the Pipelines API (without Metadata)

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 \[page 270\]](#).
- You have created a generic secret. For more information, see [Generic Secrets for Grounding \[page 271\]](#).
- You're using a supported data source and document type. For more information, see [Pipelines API \[page 292\]](#).

## Context

### → 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

1. Send a POST request to the endpoint \$AI\_API\_URL/v2/lm/document-grounding/pipelines.  
**The following examples show requests for each data source:**

### Microsoft SharePoint

The following shows an example of a SharePoint site to be indexed.

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
<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.

## AWS S3

The following shows an example of an AWS S3 repository to be indexed:

Populate the sample code with the following values:

Value	
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{{access_token}}</code>	<code>{{access_token}}</code>
<code>&lt;generic secret name&gt;</code>	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" ]
    }
  }
}'
```

## SFTP

The following shows an example of an SFTP repository to be indexed:

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
<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"]
    }
  }
}'
```

## SAP Build Work Zone

The following shows an example of a SAP Build Work Zone site to be indexed.

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
<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": "SAP Build Work Zone",
  "metadata": {
    "destination": "<generic secret name>"
  }
}'
```

## SAP Document Management Service

The following shows an example of a SAP Document Management service site to be indexed.

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
<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>"
  }
}'
```

## 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 \[page 332\]](#) or [Using the Grounding Module \[page 337\]](#).

## 7.2.1.2.1.2.1.2 Create a Document Grounding Pipeline Using the Pipelines API (with Metadata)

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 \[page 270\]](#).
- You have created a generic secret. For more information, see [Generic Secrets for Grounding \[page 271\]](#).
- You're using a supported data source and document type. For more information, see [Pipelines API \[page 292\]](#).

### Context

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 \(without Metadata\) \[page 293\]](#).

→ 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

1. Send a POST request to the endpoint `$AI_API_URL/v2/lm/document-grounding/pipelines`.  
The following examples show requests for each data source:

### Microsoft SharePoint

The following shows an example of a SharePoint site to be indexed with metadata.

Populate the sample code with the following values:

Value	
<code>{{resource_group}}</code>	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
<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>"
  }
}'
```

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.

To make your pipeline searchable, add dataRepositoryMetadata to the metadata field. For example:

```
...
  "metadata": {
    "dataRepositoryMetadata": [
      {
        "key": "example",
        "value": [
          "demo"
        ]
      }
    ]
  }
...
```

For more information, see [Search a Pipeline \[page 313\]](#).

## AWS S3

The following shows an example of an AWS S3 repository to be indexed with metadata:

Populate the sample code with the following values:

	Value
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{access_token}</code>	Your access token for SAP AI Core
<code>&lt;generic secret name&gt; (configuration)</code>	The name of the generic secret created for AWS S3
<code>&lt;generic secret name&gt; (metadata)</code>	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>"
  }
}'
```

The `metadata` attribute is optional. It accepts the destination name, which is used to connect to the AWS S3 metadata server to retrieve metadata for document indexing.

To make your pipeline searchable, add `dataRepositoryMetadata` to the `metadata` field. For example:

```
...
  "metadata": {
    "dataRepositoryMetadata": [
      {
        "key": "example",
        "value": [
          "demo"
        ]
      }
    ]
  }
...
...
```

For more information, see [Search a Pipeline \[page 313\]](#).

## SFTP

The following shows an example of an SFTP repository to be indexed with metadata:

Populate the sample code with the following values:

	Value
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{{access_token}}</code>	Your access token for SAP AI Core
<code>&lt;generic secret name&gt; (configuration)</code>	The name of the generic secret created for SFTP
<code>&lt;generic secret name&gt; (metadata)</code>	The name of the generic secret created for the SFTP Meta-Data 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.

To make your pipeline searchable, add `dataRepositoryMetadata` to the `metadata` field. For example:

```
...
  "metadata": {
    "dataRepositoryMetadata": [
      {
        "key": "example",
        "value": [
          "demo"
        ]
      }
    ]
  }
```

```
        }
    ]
...
}
```

For more information, see [Search a Pipeline \[page 313\]](#).

## 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 \[page 332\]](#) or [Using the Grounding Module \[page 337\]](#).

### 7.2.1.2.1.1.2.1.3 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/v2/1m/document-grounding/pipelines`

You can use the endpoint `$AI_API_URL/v2/1m/document-grounding/pipelines` to retrieve all document grounding pipelines.

Populate the code snippet with the following:

- `{resource_group}`: the AI resource group ID assigned to your account.
- `AI_API_URL`: the base URL of your SAP AI Core environment.
- `{access_token}`: Your access token for SAP AI Core

### 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.

The available statuses for pipelines are the following:

- 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.

For example:

```
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}}'
```

## 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 (for 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/v2/1m/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/v2/1m/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.

The available statuses for pipelines are the following:

- 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.

The available statuses for pipelines are the following:

- 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.

## 7.2.1.2.1.2.1.4 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
<code>\$top</code>	Integer	Limits the number of results returned. For example, <code>\$top=10</code> returns 10 results.
<code>\$skip</code>	Integer	Skips a specific number of results. For example, <code>\$skip=20</code> skips the first 20.
<code>\$count</code>	Boolean	If <code>true</code> , includes the total count in the response. If <code>false</code> , 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}}'
```

The available statuses for executions are the following:

- 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}}'
```

The available statuses for executions are the following:

- NEW
- UNKNOWN
- INPROGRESS

- FINISHED
- FINISHEDWITHERRORS
- TIMEOUT

## 7.2.1.2.1.2.1.5 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}}'
```

The available statuses for documents are the following:

- TO\_BE\_PROCESSED
- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED\_TO\_BE\_RETRYED
- 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}}'
```

The available statuses for documents are the following:

- TO\_BE\_PROCESSED
- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED\_TO\_BE\_RETRYED
- 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}}'
```

The available statuses for documents are the following:

- TO\_BE\_PROCESSED
- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED\_TO\_BE\_RETRYED
- 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}}'
```

The available statuses for documents are the following:

- TO\_BE\_PROCESSED

- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED\_TO\_BE\_RETRYED
- 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}}'
```

The available statuses for documents are the following:

- TO\_BE\_PROCESSED
- INDEXED
- REINDEXED
- DEINDEXED
- FAILED
- FAILED\_TO\_BE\_RETRYED
- TO\_BE\_SCHEDULED

## 7.2.1.2.1.1.2.1.6 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

## ↔ 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}}"
}'
```

## Results

The response includes a response code. Code 202 is a successful response.

### 7.2.1.2.1.1.2.1.7 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 \(with Metadata\) \[page 298\]](#).

#### 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": "value"
    }
  ]
}'
```

```

        "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": "<sname>"
                }
            }
        }
    ]
}

```

## 7.2.1.2.1.2.2 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 chunk is a small segment of a document's text, used in the Vector API to improve the accuracy and efficiency of semantic search.

## 7.2.1.2.1.2.2.1 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 \[page 270\]](#).

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 \[page 316\]](#).
2. Insert documents and chunks into the collection. For more information see [Create a Document \[page 323\]](#).

### 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 \[page 332\]](#) or [Using the Grounding Module \[page 337\]](#).

Alternatively, you can search specific collections using vector search. For more information, see [Vector Search \[page 330\]](#)

## 7.2.1.2.1.2.2.1.1 Collections

### 7.2.1.2.1.2.2.1.1.1 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 \[page 270\]](#)

#### 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:

	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
<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

#### ① Note

The `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 '{
  "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 \[page 320\]](#).

## 7.2.1.2.1.1.2.2.1.1.2      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.

### 7.2.1.2.1.1.2.2.1.1.3 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.

### 7.2.1.2.1.1.2.2.1.1.4 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.

### 7.2.1.2.1.1.2.2.1.1.5 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 the status of your collection creation.

## Next Steps

After the collection has been created successfully, you can add documents to your collection. For more information, see [Create a Document \[page 323\]](#).

### 7.2.1.2.1.1.2.2.1.1.6 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.

### 7.2.1.2.1.1.2.2.1.1.7 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/v2/1m/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/v2/1m/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 the status of your collection deletion.

## 7.2.1.2.1.2.2.1.2 Documents

### 7.2.1.2.1.2.2.1.2.1 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 \[page 314\]](#) and [Create a Collection \[page 316\]](#).

You know the ID of the collection that you want to add documents to. For more information, see [Get a Collection \[page 319\]](#).

#### Procedure

- 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
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{{access_token}}</code>	Your access token for SAP AI Core
<code>Content-Type</code>	<code>application/json</code>
<code>{collectionId}</code>	The ID of the collection
<code>metadata &lt;key&gt; &lt;value&gt; pair</code>	Your choice of metadata key value pair
<code>chunks</code>	Text content in chunks
<code>metadata &lt;key&gt; &lt;value&gt; pair (chunks)</code>	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 \[page 330\]](#).

To search for relevant chunks across all collections based on a user query, use the Retrieval API. For more information, see [Retrieval API \[page 332\]](#).

To use your documents as part of an orchestration workflow, see [Using the Grounding Module \[page 337\]](#).

## 7.2.1.2.1.2.2.1.2.2 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
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{{access_token}}</code>	Your access token for SAP AI Core
<code>Content-Type</code>	<code>application/json</code>
<code>{{collectionId}}</code>	The ID of the collection
<code>{{documentId}}</code>	The ID of the document
metadata <code>&lt;key&gt; &lt;value&gt;</code> pair	Your choice of metadata key value pair
chunks	Text content in chunks
metadata <code>&lt;key&gt; &lt;value&gt;</code> 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": "example",
          "value": "value"
        }
      ]
    }
  ]
}'
```

```
        "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 response 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

## 7.2.1.2.1.1.2.2.1.2.3 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.

## 7.2.1.2.1.2.2.1.2.4 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
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{{access_token}}</code>	Your access token for SAP AI Core
<code>{{collectionId}}</code>	The ID of the collection
<code>{{documentId}}</code>	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.

## 7.2.1.2.1.2.2.1.2.5 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	
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{{access_token}}</code>	Your access token for SAP AI Core
<code>{{collectionId}}</code>	The ID of the collection
<code>{{documentId}}</code>	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

## 7.2.1.2.1.2.2.2 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.

### Procedure

Send a POST request to the endpoint: \$AI\_API\_URL/v2/1m/document-grounding/vector/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
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

#### ↔ Sample Code

```
curl --request POST \
--url $AI_API_URL/v2/1m/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": []
    }
  ]
}'
```

```
}
```

## Results

The response is returned in JSON format, and includes relevant chunks based on similarity search..

## 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
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

Attribute	Type	Required	Description	Constraints
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

### 7.2.1.2.1.1.2.3 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 \[page 314\]](#).

#### 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](#).

### 7.2.1.2.1.1.2.3.1 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/v2/1m/document-grounding/retrieval/dataRepositories`.

Populate the sample code with the following values:

	Value
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{{access_token}}</code>	Your access token for SAP AI Core

### ↔ Sample Code

```
curl --request GET \
--url $AI_API_URL/v2/1m/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
<code>\$top</code>	Integer	Limits the number of results returned. For example, <code>\$top=10</code> returns 10 results.
<code>\$skip</code>	Integer	Skips a specific number of results. For example, <code>\$skip=20</code> skips the first 20.
<code>\$count</code>	Boolean	If <code>true</code> , includes the total count in the response. If <code>false</code> , 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}}'
```

### 7.2.1.2.1.1.2.3.2 Retrieve a Specific Data Repository

#### 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
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{{access_token}}</code>	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.

## 7.2.1.2.1.2.3.3 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
<code>{{resource_group}}</code>	The AI resource group assigned to your account
<code>\$AI_API_URL</code>	The base URL of your SAP AI Core environment. This can also be set as an environment variable.
<code>{{access_token}}</code>	Your access token for SAP AI Core
<code>&lt;user query&gt;</code>	The query that you want to be answered from the vector database
<code>dataRepositories</code>	'*' searches through all collections. Alternatively, enter a collection ID to retrieve a chunk from that collection.
<code>dataRepositoryType</code>	Supported repository types are <code>vector</code> and <code>help.sap.com</code> . Data from <code>help.sap.com</code> 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.

## 7.2.1.2.1.2 Using the Grounding Module

### Prerequisites

You have a running orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

### 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.
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_template.prompt.template	Your prompt template
config.modules.prompt_template.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-binary '{
    "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_tokens": 200,
                        "temperature": 0.2
                    }
                }
            }
        },
        "placeholder_values": {
            "groundingRequest": "<grounding query>"
        }
    }
}'
```

## Related Information

Leveraging Orchestration Capabilities to Enhance Responses 

Libraries and SDKs [page 573]

### 7.2.1.2.1.2.1 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 orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

You have onboarded a repository. For more information, see [Generic Secrets for Grounding \[page 271\]](#).

#### 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_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_template": {
  "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>"
}
```

## 7.2.1.2.1.2.2 Retrieval Using `help.sap.com`

### Prerequisites

You have a running orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

You have onboarded a repository. For more information, see [Generic Secrets for Grounding \[page 271\]](#).

## 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_template": {
        "prompt": {
            "template": [
                {
                    "role": "user",
                    "content": "<prompt>: {{?groundingRequest}}"
                }
            ]
        },
        "model": {
            "name": "<modelName>",
            "version": "<modelVersion>",
            "params": {
                "max_tokens": 50,
                "temperature": 0.1,
                "frequency_penalty": 0,
                "presence_penalty": 0
            }
        }
    }
},
"placeholder_values": {
    "groundingRequest": "<grounding request>"
}
}

```

## 7.2.1.2.2 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.

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 \[page 393\]](#) 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 \[page 379\]](#).

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 `version`): The model version is optional and defaults to "latest".
- Timeout (parameter `timeout`): is optional and is specified in seconds. Min: 1, max 600, 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_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 \[page 423\]](#).

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_template": {
    "prompt": {
        "template_ref": {
            "id": "d22342e7-1f91-4c52-a794-04833bc2574a"
            "scope": "tenant"
        }
    }
},

```

or

```

"prompt_template": {
    "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"data:image/jpeg;base64,{base64_image}"
}
```

or

```
"image_url": {
    "url": "https://some.jpg"
}
```

## Input PDF Support

PDFs can be provided as additional input for Gemini models only.

```
{
  "config": {
    "modules": {
      "prompt_templating": {
        "model": {
          "name": "gemini-2.5-flash",
          "params": {
            "temperature": 0.1
          }
        },
        "prompt": {
          "template": [
            {
              "role": "user",
              "content": [
                {
                  "type": "image_url",
                  "image_url": {
                    "url": "data:application/pdf;base64,{?base64_pdf}"
                  }
                },
                {
                  "type": "text",
                  "text": "what does the document say?"
                }
              ]
            }
          ]
        }
      },
      "stream": {
        "enabled": false
      }
    },
    "placeholder_values": {

```

```
"base64_pdf" :  
"JVBERi0xLjQKMSAwIG9iago8PC9UeXB1IC9DYXRhbG9nCi9QYWdlcyAyIDAgUGo+PgplbmRvYmoKMiAw  
IG9iago8PC9UeXB1IC9QYWdlcwovS2lkcBbMyAwIFJdCi9Db3VudCAxCj4+CmVuZG9iagozIDAgb2JqC  
jw8L1R5cGUgL1BhZ2UKL1BhcmVudCAyIDAgUGovTWVkaWFcb3ggWzAgMCA1OTugODQyXQovQ29udGVudH  
MgNSAwIFIKL1J1c291cmN1cyA8PC9Qcm9jU2V01FsvUERGIC9UZXh0XQovRm9udCA8PC9GMSA0IDAgUj4  
+Cj4+Cj4+CmVuZG9iago0IDAgb2JqCjw8L1R5cGUgL0ZvbhQKL1N1YnR5cGUGL1R5cGUxci9OYW11IC9G  
MQovQmFzzZUZvbhQgL0hlbHZ1dG1jYQovRW5jb2RpbmcgL01hY1JvbWFuRW5jb2Rpbmcpkj4KZW5kb2JqC  
jUgMCBvYmoKPDwvTGVuZ3RoIDUzCj4+CnN0cmVhbQpCVAovrjEgMjAgVGYKjIwIDQwMCBUZAooRHvtbx  
kgUERGKSBuAgpFVaPlbmRzdHJ1YW0KZW5kb2JqChhyZWYKMCAC2cjAwMDAwMDAwMDAgNjU1MzUgZgowMDA  
wMDAwMDA5IDAwMDAwIG4KMDAwMDAwMDA2MyAwMDAwMCBuCjAwMDAwMDAxMjQgMDAwMDAgbgowMDAwMDAw  
MjC3IDAwMDAwIG4KMDAwMDAwMDM5MiAwMDAwMCBuCnRyYWlsZXIKPDwvU216ZSA2C19Sb290IDEgMCBSC  
j4+CnN0YXJ0eHJ1Zgo0OTUKJSVFT0YK"  
}  
}
```

The URL for `image_url` also a web-based URL of the PDF.

```
"image_url": {
    "url": "https://assets.anthropic.com/m/1cd9d098ac3e6467/original/Claude-3-Model-Card-October-Addendum.pdf"
}
```

### **7.2.1.2.2.1 Few-Shot Learning**

## Prerequisites

You have a running orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

## 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_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 \[page 573\]](#)

## 7.2.1.2.2 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 orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

### 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_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": "I am deeply in love with you. You are my everything."
      }
    ]
  }
}'
```

```

        "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": "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\nfor thee, a love that transcends the bounds of time and space.\n2. Thou art the\nmuse of my existence, inspiring a passion within me that resonates with the very\nessence 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\nfor thee, a love that transcends the bounds of time and space.\n2. Thou art the\nmuse of my existence, inspiring a passion within me that resonates with the very\nessence of nature itself."
            },
            "finish_reason": "stop"
        }
    ],
    "usage": {
        "completion_tokens": 55,
        "prompt_tokens": 212,
        "total_tokens": 267
    }
}
}

```

## Related Information

[Leveraging Orchestration Capabilities to Enhance Responses](#)

[Libraries and SDKs \[page 573\]](#)

### 7.2.1.2.2.3 Minimal Call

#### Prerequisites

You have a running orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

#### 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_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 [page 573]

### 7.2.1.2.2.4 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.

#### 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_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_eeld74bde0",
    "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_eeld74bde0",
    "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_template": {
"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_i13KDaSC5zm6naTOnYv5VSZT",
"type": "function",
"function": {
"name": "get_inventory_quantity",
"arguments": "{\"product_id\":123456}"
}
]
},
# append function result
{
"role": "tool",
"tool_call_id": "call_i13KDaSC5zm6naTOnYv5VSZT",
"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_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-BuZvTwr6ZSxebqgHn8vp46zntVcZ1",  
            "content": "The user wants to know the inventory quantity for product ID 123456. The system will call the get_inventory_quantity function with arguments {\"product_id\":123456}."  
        }  
    }  
}
```

```

    "object": "chat.completion",
    "created": 1752824167,
    "model": "gpt-4o-2024-08-06",
    "system_fingerprint": "fp_eeld74bde0",
    "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_eeld74bde0",
    "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
    }
}
}

```

### 7.2.1.2.3 Translation

The 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.

The output translation module is optional. It allows you to translate LLM output into a chosen target language.

To use 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. 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 \[page 361\]](#)

[Enhance Model Consumption with Output Translation \[page 363\]](#)

### 7.2.1.2.3.1 Enhance Model Consumption with Input Translation

To configure translation, add a `translation` section to the `modules` of your config.

For example:

#### ↔ Sample Code

```
{
  "config": {
    "modules": {
      ...
      "translation": {
        "type": "sap_document_translation",
        "config": {
          "source_language": "de-DE",
          "target_language": "en-US"
        }
      }
      ...
    }
  }
}
```

The `translation` 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

```
{
  "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. \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}' '"
          }
        ]
      },
      "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: 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!"
    }
  }
}
```

## 7.2.1.2.3.2 Enhance Model Consumption with Output Translation

To configure output translation, add an `translation` section to the `modules` of your config.

For example:

### → Sample Code

```
{  
  "config": {  
    "modules": {  
      ...  
      "translation": {  
        "output": {  
          "type": "sap_document_translation",  
          "config": {  
            "source_language": "en-US",  
            "target_language": "de-DE"  
          }  
        }  
      }  
      ...  
    }  
    ...  
  }  
}
```

The `translation` 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

```
{
  "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": {
        "type": "sap_document_translation",
        "config": {
          "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!"
  }
}
```

### 7.2.1.2.4 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)
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 \[page 245\]](#).

#### ⚠ 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.

### 7.2.1.2.4.1 Enhancing Model Consumption with Data Masking

#### Prerequisites

You have a running orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

#### 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/v2/completion \
--header 'content-type: application/json' \
```

```

--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP" \
--data-raw '{
  "config": {
    "modules": {
      "prompt_template": {
        "prompt": {
          "template": [
            {
              "role": "user",
              "content": "{{?input}}"
            }
          ]
        },
        "model": {
          "name": "gpt-4o",
          "params": {
            "max_tokens": 300,
            "temperature": 0.1,
            "frequency_penalty": 0,
            "presence_penalty": 0
          }
        }
      },
      "masking": {
        "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"
    }
  }
}'

```

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.

#### 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": "Input to LLM is masked successfully.",
      "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_eeld74bde0",
        "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_eeld74bde0",
    "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
    }
}
}

```

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": "Input to LLM is masked successfully.",
    "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_eeld74bde0",
    "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_eeld74bde0",
    "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": {
        "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.

## Related Information

[Leveraging Orchestration Capabilities to Enhance Responses](#)

[Libraries and SDKs \[page 573\]](#)

### 7.2.1.2.5 Content 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. 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

### Harm Categories

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.

#### ⓘ 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

When multiple content filtering types are configured together, each type is processed as a separate request. Costs will be incurred for each filtering type applied.

## 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 have not been requested by the user

The orchestration service doesn't take these violations into account when processing Llama Guard responses, and only filters violations in categories that have been requested by the user.

## Related Information

[Enhancing Model Consumption with Input Filtering \[page 248\]](#)

[Enhancing Model Consumption with Output Filtering \[page 376\]](#)

## 7.2.1.2.5.1 Enhancing Model Consumption with Input Filtering

### Prerequisites

You have a running orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

### Process

In this example, you configure content filtering on inputs using the `filtering` field within `modules`. 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`.

For `azure_content_safety`, you can set different scores for each of the four 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 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/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-mini",
          "params": {
            "max_tokens": 300,
            "temperature": 2
          }
        }
      },
      "filtering": {
        "input": {
          "filters": [

```

```

{
  "type": "azure_content_safety",
  "config": {
    "hate": 0,
    "violence": 2,
    "sexual": 4,
    "self_harm": 6,
    "prompt_shield": 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
  }
}
]
}
},
"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, 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.

```

{
  "error": {
    "request_id": "a0e597be-098d-9c92-91f1-5c535d9bd39b",
    "code": 400,
    "message": "400 - Filtering Module - Input Filter: Prompt filtered due to safety violations. Please modify the prompt and try again.",
    "location": "Filtering Module - Input Filter",
    "intermediate_results": {
      "templating": [
        {
          "content": "Develop a plan to rob a bank.",
          "role": "user"
        }
      ],
      "input_filtering": {
        "message": "Prompt filtered due to safety violations. Please modify the prompt and try again.",
        "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,
        "sex_crimes": false
    }
}
}
```

## Related Information

Leveraging Orchestration Capabilities to Enhance Responses   
Libraries and SDKs [page 573]

## 7.2.1.2.5.2 Enhancing Model Consumption with Output Filtering

## Prerequisites

You have a running orchestration deployment and have retrieved your orchestration deployment URL. For more information, see [Get Your Orchestration Deployment URL \[page 128\]](#) and [Create a Deployment for Orchestration \[page 421\]](#).

## 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/v2/completion \
--header 'content-type: application/json' \
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP" \
--data-raw '{
  "config": {
```

```

"modules": {
    "prompt_template": {
        "prompt": {
            "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}}"
                }
            ]
        },
        "model": {
            "name": "<model-name>",
            "version": "latest",
            "params": {
                "max_tokens": 50,
                "temperature": 0.1,
                "frequency_penalty": 0,
                "presence_penalty": 0
            }
        }
    },
    "filtering": {
        "input": {
            "filters": [
                {
                    "type": "azure_content_safety",
                    "config": {
                        "hate": 2,
                        "self_harm": 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,
                        "self_harm": 0,
                        "sexual": 0,
                        "violence": 0
                    }
                }
            ]
        }
    }
}

```

```

        }
    },
},
"placeholder_values": {
    "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": "4435c6a6-f08e-9561-9a58-1103ac581550",
    "intermediate_results": [
        "templating": [
            {
                "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. Homicides happen weekly",
                "role": "user"
            }
        ],
        "output_filtering": {
            "message": "1 of 1 choices failed the output filter.",
            "data": {
                "choices": [
                    {
                        "index": 0,
                        "azure_content_safety": {
                            "Hate": 0,
                            "SelfHarm": 0,
                            "Sexual": 2,
                            "Violence": 0
                        }
                    }
                ]
            }
        }
    ],
    "llm": {
        "id": "chatcmpl-Bua9DAmEQjcNASrZL4oWMOrPVlGPm",
        "object": "chat.completion",
        "created": 1752825019,
        "model": "gpt-4o-mini-2024-07-18",
        "system_fingerprint": "fp_efad92c60b",
        "choices": [
            {
                "index": 0,
                "message": {
                    "role": "assistant",
                    "content": "***Sublet Available in Downtown Area!**\n\nLooking for a temporary home in the heart of downtown? I'm subletting my cozy 1-bedroom apartment, perfect for anyone wanting to experience city life. The apartment features a spacious living area, modern kitchen, and is conveniently located near public transport, shops, and restaurants. Available from [start date] to [end date]. \n\nRent: $[amount] per month, utilities included. \n\nContact me for more details or to schedule a viewing!\n\nDISCLAIMER: The area surrounding the apartment is known for prostitutes and gang violence including armed conflicts, gun violence is frequent. Homicides happen weekly."
                }
            }
        ]
    }
}

```

```

        "finish_reason": "stop"
    },
],
"usage": {
    "completion_tokens": 131,
    "prompt_tokens": 70,
    "total_tokens": 201
}
},
"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": 131,
        "prompt_tokens": 70,
        "total_tokens": 201
    }
}
}

```

### ⚠ 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 \[page 573\]](#)

### 7.2.1.3 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 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.

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 is 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"  
        }  
      ]  
    }  
  ]  
}
```

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` will be mapped to Vertex AI's `maxOutputTokens`. Additionally, any model-specific parameters can be sent. However when using model specific parameters they should 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 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
    }
}

```

## 7.2.1.4 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 \[page 418\]](#).

Send POST request to the endpoint `{Orchestration URL}/v2/embeddings` and populate the sample code with the following values:

Value	
<code>{{orchestration-deployment-url}}</code>	The URL of your RUNNING orchestration deployment.
<code>{{token}}</code>	Your authorization token for SAP AI Core
<code>{{resource_group}}</code>	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 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 \[page 237\]](#) and [Model Configuration \[page 251\]](#).

```
curl --request POST \
--url {{Orchestration 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": {
        "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!" ,  
        "type": "text"  
    }'  
}'
```

## 7.2.1.5 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. Benchmarking 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 \[page 10\]](#) 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:

### ↔ Sample Code

```
{  
  "orchestration_config": {  
    "stream": true,  
    "module_configurations": {  
      ...  
    },  
    "input_params": {  
      ...  
    }  
  }  
}
```

You can configure streaming behavior in the following ways:

- Chunk Size

- Delimiters

## Configure Streaming Using Chunk Size

The `chunk_size` parameter defines the maximum 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.

### Sample Code

```
{
  "orchestration_config": {
    "stream": true,
    "stream_options": {
      "chunk_size": 200
    },
    "module_configurations": {
      ...
      "filtering_module_config": {
        "output": {
          "filters": [
            ...
          ],
          "stream_options": {
            "overlap": 10
          }
        }
      }
      ...
    },
    "input_params": {
      ...
    }
  }
}
```

## 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.

### Sample Code

```
{
```

```
"orchestration_config": {  
    "stream": true,  
    "stream_options": {  
        "delimiters": [".", "!", "?"]  
    },  
    "module_configurations": {  
        ...  
    },  
    "input_params": {  
        ...  
    }  
}
```

## 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_PERSON_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": {  
            ...  
        }  
    }  
}
```

## 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].

### 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": "You are beautiful."
          }
        }
      ]
    }
}
```

```
        "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 an Error Response

#### ↔ Sample Code

```
data: {"json": "payload"}  
data: {"code": 500, "message": "message"}
```

## 7.2.1.6 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 `modelName`s and `modelVersion`s 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"  
    }  
  ]  
}
```

### 7.2.1.7 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](#).

## 7.2.1.8 Error Handling

If an error occurs, the response will contain an error code and message. The following example shows a request that is missing a parameter in the templating module configuration.

```
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"
    }
  }
}'
```

In this case, the response from the service contains an error code in the `code` field, a `message`, and a `location` indicating which orchestration module encountered the error. In all error cases, the `module_results` field only contains results for modules that successfully finished. In this example, the `module_results` are empty because the first module in the pipeline encountered the error.

```
{
  "request_id": "3e988846-1360-4a4a-a7ad-e77b85057321",
  "code": 400,
  "message": "Missing required parameters: ['phrase']",
  "location": "Module: Templating",
  "module_results": []
}
```

## 7.2.1.9 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 \[page 573\]](#)

## 7.2.2 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 \[page 127\]](#) and [Create a Deployment for a Generative AI Model \[page 395\]](#).

The following scenarios are available:

Global Scenario	Executable ID	Description
<code>foundation-models</code>	<code>aicore-cohere</code>	Cohere models hosted and accessed through SAP AI Core.
<code>foundation-models</code>	<code>aicore-ibm</code>	Models from IBM that are hosted and managed by SAP and accessed through SAP AI Core.
<code>foundation-models</code>	<code>aicore-mistralai</code>	Models from Mistral AI that are hosted and managed by SAP and accessed through SAP AI Core.
<code>foundation-models</code>	<code>aicore-nvidia</code>	Models from Nvidia that are hosted and managed by SAP and accessed through SAP AI Core.
<code>foundation-models</code>	<code>aicore-opensource</code>	Open source models that are hosted and managed by SAP and accessed through SAP AI Core.
<code>foundation-models</code>	<code>aws-bedrock</code>	AWS Bedrock provides access to Foundation models from Anthropic, Amazon and other providers.
<code>foundation-models</code>	<code>azure-openai</code>	The Azure OpenAI Service provides REST API access to OpenAI's LLMs.
<code>foundation-models</code>	<code>gcp-vertexai</code>	GCP Vertex AI provides access to PaLM 2 and Gemini 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 Generative AI Models \[page 399\]](#).

## 7.2.2.1 Get an Auth Token

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

### Prerequisites

curl is likely to be installed on your operating system by default. To check, open a command prompt and enter `curl -v`. If curl isn't installed, download and install it from <https://curl.se/>.

#### ⓘ Note

On macOS, you may need to install jq so that you can follow the curl commands.

1. Install brew from <https://brew.sh/>.
2. In a Terminal session, run `brew install jq` to install jq in your shell environment.

### 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 auth token by sending the following request:

```
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...
```

## 7.2.2.2 Create a Deployment for a Generative AI Model

### 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 \[page 64\]](#) and [Upgrade a Service Plan \[page 67\]](#).

### 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.

You can view the available models and their details by sending a GET request to the endpoint  
`{apiurl}/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"
          ],
          "deploymentUrl": "https://api.saphana.ai/v2/lm/scenarios/foundation-models/gpt-4o/versions/1/deployments/1"
        }
      ]
    }
  ]
}
```

```

    "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

# Using the API

## Procedure

- 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, using “latest” will use the latest version of the model available in SAP AI Core.
- Where model version is not listed, it is not applicable.

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

- 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 \[page 90\]](#).

```
curl --location '{apiurl}/v2/lm/scenarios' \
--header 'AI-Resource-Group: default' \
--header "Authorization: Bearer $AUTH_TOKEN"
```

The scenarios listed contain a scenario with the id `foundation-models`.

- 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.

You receive a unique `configurationId` in the response.

4. 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"
}'
```

5. 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"
```

## Next Steps

When the deployment is running, the model can be accessed using the `deploymentUrl` provided in the response. For more information, see [Consume Generative AI Models \[page 399\]](#).

## Model Lifecycle

Model versions have deprecation dates. Where a model version is specified in a deployment, the deployment will stop working on the deprecation date of that model version.

Implement one of the following model upgrade options:

- **Auto Upgrade:** Create a new 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 new generative AI configuration with your chosen replacement `modelVersion` and use it to patch your deployment. This model version will be 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`.

### 7.2.2.3 Consume Generative AI 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 for a Generative AI Model \[page 395\]](#).

#### 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 does not take any responsibility for the quality of the content in the input to or output of the underlying generative AI models. This includes but is not limited to bias, hallucinations, or inaccuracies. The user is responsible for verifying the content.

 Caution

Do not store personally identifiable information in prompts when using the 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.

## Procedure

Consume your generative AI deployment using your deployment URL and the example payloads provided. For more information, see the following:

- [Example Payloads for Inferencing - SAP AI Core Hosted \[page 408\]](#) for models with `executionIds` indicating a third party provider.
- [Example Payloads for Inferencing - Remote Models \[page 400\]](#) for models with `executionIds` indicating that a model is hosted on SAP AI Core (beginning with `aicore-`).

### 7.2.2.3.1 Example Payloads for Inferencing - Remote Models

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`.

Ensure that you have the following headers set:

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 <a href="#">Create a Deployment for a Generative AI Model [page 395]</a> . Alternatively, you can replace the <code>\$DEPLOYMENT_URL</code> placeholder in the curl command with your deployment URL.

For more information about supported parameters, see [Supported Parameters \[page 417\]](#).

If you want to remove a model, delete its deployment. For more information, see [Delete a Single Deployment \[page 545\]](#) and [Delete Multiple Deployments \[page 547\]](#).

## Open AI

The `executableId` is `azure-openai`.

## Completions

For more information from the model provider, see [Microsoft Azure Open AI Chat Completions](#).

**gpt-4-32k | gpt-4 | gpt-3.5-Turbo-16k | gpt-3.5-Turbo | o4-mini | o3 | gpt-4.1 | gpt-4.1-mini | gpt-4.1-nano | gpt-5 | gpt-5-mini | gpt-5-nano**

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-4-Turbo | 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://example.com/picture.jpg"
                    }
                }
            ]
        }
    ]
}'
```

```
        "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"
}'
```

## Vertex AI

The executableId is gcp-vertexai.

## Gemini

For more information from the model provider, see [Vertex AI Gemini](#).

### **Gemini 1.0 Pro**

```
curl --location '$DEPLOYMENT_URL/models/gemini-1.0-pro:generateContent' \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $AUTH_TOKEN" \
--data '{
    "contents": [
        {
            "role": "user",
            "parts": {
                "text": "Hello!"
            }
        },
        {
            "role": "model",
            "parts": {
                "text": "Argh! What brings ye to my ship?"
            }
        }
    ]
}'
```

```

    {
        "role": "user",
        "parts": [
            {
                "text": "Wow! You are a real-life pirate!"
            }
        ],
        "safety_settings": {
            "category": "HARM_CATEGORY_SEXUALLY_EXPLICIT",
            "threshold": "BLOCK_LOW_AND ABOVE"
        },
        "generation_config": {
            "temperature": 0.9,
            "topP": 1,
            "candidateCount": 1,
            "maxOutputTokens": 2048
        }
    }
}

```

## Gemini 1.5 Pro

### Text Input

```

curl --location '$DEPLOYMENT_URL/models/gemini-1.5-pro: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/gemini-1.5-
pro: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."
                }
            ]
        }
    ]
}'

```

## Gemini 1.5 Flash | Gemini 2.0 Flash | Gemini 2.0 Flash Lite

### 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."
      }
    ]
  }
}'
```

### 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

## AWS Bedrock

The executableId is aws-bedrock.

## Anthropic Claude

For more information from the model provider, see [AWS Bedrock Claude](#).

### Claude 3 Sonnet | Claude 3.5 Sonnet | Claude 3 Opus | Claude 3 Haiku | Claude 4 Sonnet | Claude 4 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"
        ],
        "temperature": 0.7
    },
    "messages": [
        {
            "content": [
                {
                    "text": "Perplexity means?"
                }
            ]
        }
    ]
}'
```

```
        ],
        "role": "user"
    }
}
}'
```

## Claude 3.7 Sonnet

```
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"
        }
    ]
}'
```

## Amazon Titan

For more information from the model provider, see [AWS Bedrock Titan](#).

### Titan Text Express | Titan Text Lite

```
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 am AI?",
    "textGenerationConfig": {
        "maxTokenCount": 10,
        "stopSequences": [],
        "temperature": 0,
        "topP": 1
    }
}'
```

### 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/  
4QDeRXhpZgAASUkqAAgAAAAGABIBAwABAAAAAQAAABoBBQABAAA AVgAAABsBBQABAAAAXgAACgBAwABA  
AAA AgAAABMCAwABAAAAAQAAAGmHBAABAAA AZgAAAAAA ABIAAAAAQAAAEgAAAABAAA ABwAAkAcABAAA AD  
AyMTABkQcABAAAA AECAwCGkgcAFgAAAMAAAAAA oAcABAAA ADxMDABoAMAAQAAAP//  
AAACoAQAAQAAAMgAAAADoAQAAQAAAMgAAAAAAA QVNDSUkAAABQaWNzdW0gSUQ6IDEyM//  
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 for Amazon Bedrock models can be invoked by replacing the deployment URL with `$DEPLOYMENT_URL/invoke-with-response-stream`.

### 7.2.2.3.2 Example Payloads for Inferencing - SAP AI Core Hosted

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`.

Ensure that you have the following headers set:

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 <a href="#">Create a Deployment for a Generative AI Model [page 395]</a> .  Alternatively, you can replace the <code>\$DEPLOYMENT_URL</code> placeholder in the curl command with your deployment URL.

For more information about supported parameters, see [Supported Parameters \[page 417\]](#).

If you want to remove a model, delete its deployment. For more information, see [Delete a Single Deployment \[page 545\]](#) and [Delete Multiple Deployments \[page 547\]](#).

## Open Source

The executableId is `aicore-opensource`

## Meta Llama 3.1 70b instruct | Mistral Mixtral 8x7b instruct v01

For more information from the model providers, see [Meta Llama](#) and [Mistral](#).

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": "Sample prompt"
        }
    ],
    "max_tokens": 100
}'
```

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
}'
```

## Mistral

The executableId is aicore-mistralai

For more information from the model provider, see [Mistral Large](#) and [Mistral Small](#).

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

### 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": {
        "content": " Accountability and responsibility are related but distinct concepts.\n\n**Responsibility** is about the tasks and duties assigned to you. It's what you're expected to do as part of your role or agreement. For example, a project manager is responsible for planning and overseeing a project. Responsibility is often proactive and focuses on what you should do.\n\n**Accountability**, on the other hand, is about answering for the outcomes of your responsibilities. It",
        "role": "assistant",
        "tool_calls": []
      },
      "stop_reason": null
    }
  ],
  "created": 1730096376,
  "id": "<id>",
  "model": "<modelName>",
  "object": "chat.completion",
  "usage": {
    "completion_tokens": 100,
    "prompt_tokens": 23,
    "total_tokens": 123
  }
}
```

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-1-1024x683.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 [Images \[page 13\]](#).

## 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
}
```

## IBM

The executableId is aicore-ibm

## Granite

For more information from the model provider, see [IBM Granite](#).

### ibm--granite-13b-chat

```
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": {
                "content": "Accountability and responsibility are two related but distinct concepts that are often used interchangeably, but they have different nuances. Accountability refers to being answerable or responsible"
            }
        }
    ]
}
```

for something, typically for an outcome, decision, or action. It implies that there is a level of ownership and control over the result, and one is expected to take responsibility for the consequences of their actions. Accountability can be held by individuals, teams, or organizations, and it often involves being transparent, communicative, and responsible",

```

        "role": "assistant",
        "tool_calls": []
    },
    "stop_reason": null
}
],
"created": 1730097489,
"id": "<id>",
"model": "<modelName>",
"object": "chat.completion",
"usage": {
    "completion_tokens": 100,
    "prompt_tokens": 24,
    "total_tokens": 124
}
}
```

## NVIDIA

The executableId is aicore-nvidia

## Granite

For more information from the model provider, see [Nvidia](#).

### **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"
}'
```

## Cohere

The executableId is aicore-cohere

## Reasoning

### Command A Reasoning

For more information from the model provider, see [Cohere Comman A Reasoning](#).

```
curl --location "$DEPLOYMENT_URL/v2/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?"
        }
    ]
}'
```

## Perplexity

The executableId is perplexity-ai

## Completions

### sonar | sonar-pro

↔ Sample Code

```
curl --location '$DEPLOYMENT_URL/chat/completions' \
--header 'AI-Resource-Group: default' \
--header 'Content-Type: application/json' \
--header "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?"
        }
    ]
}'
```

### 7.2.2.3.3 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 frequent 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
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

## 7.2.2.4 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 \[page 573\]](#)

## 7.3 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.

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.

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

## 7.4 Additional Concepts

### 7.4.1 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 Your Orchestration Deployment URL \[page 128\]](#) or [Model Restriction \[page 389\]](#).

- 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 \[page 64\]](#) and [Upgrade a Service Plan \[page 67\]](#).
- You have completed the client authorization for your preferred user interface. For more information, see [Use a Service Key \[page 75\]](#).

#### 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 \[page 573\]](#).

#### 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/v2/1m/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 "$AI_API_URL/v2/1m/deployments/$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"
  }
}

```

- 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>
```

## 7.4.2 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 \[page 423\]](#).
- Orchestration configs. For more information, see [Orchestration Config Management \[page 435\]](#).

### 7.4.2.1 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 \(Imperative\) \[page 424\]](#).

- 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 \(Declarative\) \[page 426\]](#).

The interfaces can all be consumed either directly via the API or through Orchestration.

### 7.4.2.1.1 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 does not 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 `{{apiurl}}/v2/lm/promptTemplates`.

##### ↔ Sample Code

```
curl -X POST "{{apiurl}}/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 \[page 429\]](#).

## 7.4.2.1.2 Create a Prompt Template (Declarative)

### Prerequisites

- You've created and synced a git repository. For more information, see [Add a Git Repository \[page 80\]](#).
- You've created an application that points to the prompt templates in your git repository. For more information, see [Create an Application \[page 86\]](#).

### 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.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 `{apiurl}/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>`. Declarative 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 \[page 430\]](#).

You can change templates locally and push them to git.

### → Tip

For declaratively managed templates, use the history capabilities in git to track your changes.

### 7.4.2.1.3 Get a Prompt Template

#### Prerequisites

- You have created a prompt template. For more information, see [Create a Prompt Template \(Imperative\) \[page 424\]](#) and [Create a Prompt Template \(Declarative\) \[page 426\]](#).

#### 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 \[page 221\]](#).

##### ⓘ 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 does not need to be provided in the headers.

#### Procedure

Send a GET request to the endpoint `{apiurl}/v2/lm/promptTemplates`, and include the information for your chosen retrieval method.

##### ↪ Sample Code

```
curl -X GET "{apiurl}/v2/lm/promptTemplates?name=example-prompt-  
template&scenario=categorization&version=0.0.1" \  
-Header "Authorization: Bearer <your_auth_token>"
```

#### ↔ Sample Code

```
curl -X GET "{{apiurl}}/v2/lm/promptTemplates/{{promptTemplateId}}" \
-Header "Authorization: Bearer <your_auth_token>"
```

### 7.4.2.1.4 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 does not need to be provided in the headers.

#### Procedure

Send a GET request to endpoint {{apiurl}}/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 "{{apiurl}}/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"
    }
}
]
}
}

```

### 7.4.2.1.5 Use a Prompt Template

#### Prerequisites

- You have created a prompt template. For more information, see [Create a Prompt Template \(Imperative\) \[page 424\]](#).

#### 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 does not 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 `{apiurl}/v2/lm/promptTemplates/{promptTemplateId}/substitution` and add your variable values in the body.
- ⓘ Sample Code

```
curl -X POST "{apiurl}/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 `{apiurl}/v2/lm/scenarios/{scenarioId}/promptTemplates/{promptTemplateName}/versions/{versionId}/substitution` and add your variable values in the body.

### ⓘ Sample Code

```
curl -X POST "{apiurl}/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>`

## 7.4.2.1.6 Import a Prompt Template

### Prerequisites

- You have created and exported a prompt template. For more information, see [Create a Prompt Template \(Imperative\) \[page 424\]](#) and [Export a Prompt Template \[page 433\]](#).

### 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 does not 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 {{apiurl}}/v2/lm/promptTemplates/import, and include the name of your file in the file parameter.

#### ↔ Sample Code

```
curl -X POST "{{apiurl}}/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"
        }
    }
}

```

### 7.4.2.1.7 Export a Prompt Template

#### Prerequisites

- You have created a prompt template. For more information, see [Create a Prompt Template \(Imperative\) \[page 424\]](#).

#### 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 does not need to be provided in the headers.

## Procedure

Send a GET request to the endpoint {{apiurl}}/v2/lm/promptTemplates/{{promptTemplateId}}/export, and include the name of your file in the output parameter.

### ↗ Sample Code

```
curl -X GET "{{apiurl}}/v2/lm/promptTemplates/{{promptTemplateId}}/export" \
-H "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
```

### 7.4.2.1.8 Delete a Prompt Template

#### Prerequisites

- You have created a prompt template. For more information, see [Create a Prompt Template \(Imperative\) \[page 424\]](#).
- You know the template ID of the prompt template that you want to delete. For more information, see [Get a Prompt Template \[page 428\]](#).

## 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 does not 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 {{apiurl}}/v2/lm/promptTemplates/{{promptTemplateId}} and include the template ID of the prompt you want to delete.

### ↴ Output Code

```
{  
  "message": "Prompt deleted successfully."  
}
```

## 7.4.2.2 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.

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.

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.

#### ⓘ Note

Orchestration configs are only available through the imperative API and are managed at the tenant level.

### 7.4.2.2.1 Create an Orchestration Config

#### Context

The imperative API supports full CRUD operations and is recommended for both design-time and runtime orchestration configuration 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 `{ {apiurl} }/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 "{{apiurl}}/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?"
                    }
                }
            }
        }
    }
}'
```

## Example With Prompt Template Reference by ID

### ↳ Sample Code

```
curl -X POST "{{apiurl}}/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": "a460d210-df38-4867-9535-7a556701a4b0"
                    }
                }
            }
        }
    }
}'
```

```
        }
    }
}
}'
```

## Example With Prompt Template Reference by ID

### ↔ Sample Code

```
curl -X POST "{{apiurl}}/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": "gpt-4o-mini",
                    "params": {
                        "temperature": 0.7,
                        "max_tokens": 500
                    }
                }
            }
        }
    }
}'
```

## Results

You will receive an output JSON containing details about your configuration, including an ID and success message.

### ↔ Output Code

```
{
    "id": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
    "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 \[page 439\]](#).

### 7.4.2.2.2 Get Orchestration Configs

#### Prerequisites

You have created an orchestration config. For more information, see [Create an Orchestration Config \[page 436\]](#).

#### Context

You can list all orchestration configs or retrieve a specific orchestration config by ID, or by the combination of name, scenario, and version.

Orchestration configs can also be retrieved and consumed in orchestration completion requests. For more information, see [Use an Orchestration Config in LLM Orchestration \[page 442\]](#).

##### ① Note

Retrieval by ID offers immutability; guaranteeing that the orchestration config behind the ID will not change.

Retrieval by name, scenario, and version are not immutable, and the latest iteration of the orchestration config is retrieved.

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`, 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": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
      "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": "b23cd456-ef78-9012-3456-789abcdef012",
      "name": "another-orchestration-config",
      "version": "1.0.0",
      "scenario": "data-analysis",
      "is_version_head": false
    }
  ]
}
```

```
        "creation_timestamp": "2024-08-19T10:30:45.123000",
        "managed_by": "imperative",
        "is_version_head": true
    }
]
```

### 7.4.2.2.3 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": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
      "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": "a460d210-df38-4867-9535-7a556701a4b0"
          }
        },
        "model": {
          "name": "gpt-4o-mini",
          "params": {
            "temperature": 0.7,
            "max_tokens": 500
          }
        }
      }
    ]
  }
}

```

#### 7.4.2.2.4 Use an Orchestration Config in LLM Orchestration

##### Prerequisites

You have created an orchestration config. For more information, see [Create an Orchestration Config \[page 436\]](#).

##### 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:

- **AI\_API\_URL:** the base URL of your SAP AI Core environment.
- **{ {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 "{{apiurl}}/v2/completion" \
-H "Authorization: Bearer <your_auth_token>" \
-H "Content-Type: application/json" \
-d '{
    "config_ref": {
        "id": "f47ac10b-58cc-4372-a567-0e02b2c3d479"
    },
    "placeholder_values": {
        "userQuery": "What are the benefits of SAP AI Core?",
        "inputContext": "enterprise AI platform"
    }
}'
```

## 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 "{{apiurl}}/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": "How do I integrate with SAP AI Core?",
        "inputContext": "API integration"
    },
    "messages_history": [
        {
            "role": "system",
            "content": "You are helping with technical integration questions."
        }
    ]
}'
```

## Method 3: Inline Configuration (for comparison)

You can also use orchestration configurations inline without storing them first.

```
curl -X POST "{{apiurl}}/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": "gpt-4o-mini"
                }
            }
        }
}'
```

```
        },
    },
    "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

## 7.4.2.2.5 Export an Orchestration Config

### Prerequisites

You have created an orchestration config. For more information, see [Create an Orchestration Config \[page 436\]](#).

### 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: a460d210-df38-4867-9535-7a556701a4b0
      model:
        name: gpt-4o-mini
        params:
          temperature: 0.7
          max_tokens: 500
```

## 7.4.2.2.6 Import an Orchestration Config

### Prerequisites

You have created and exported an orchestration config. For more information, see [Create an Orchestration Config \[page 436\]](#) and [Export an Orchestration Config \[page 444\]](#).

### 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": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
  "name": "example-orchestration-config",
  "version": "0.0.1",
  "scenario": "customer-support"
}
```

## 7.4.2.2.7 Delete an Orchestration Config

### Prerequisites

You have created an orchestration config. For more information, see [Create an Orchestration Config \[page 436\]](#).

You know the config ID of the orchestration config that you want to delete. For more information, see [Get Orchestration Configs \[page 439\]](#).

### 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."  
}
```

## 7.4.3 Rate Limit Management

Quota-management APIs let you check and modify rate-limits for generative AI model inference calls.

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, though this can be customized through specific configuration. Additionally, all versions of a particular model share the same configured rate-limit allocation.

You can use the following API endpoints to check your quota limits and request quota limit updates.

Ensure that you have the following headers set:

Header	Value
Authorization	Bearer \$AUTH_TOKEN
AI_API_URL	Your deployment URL

### Get Rate Limits

Send a GET request to the endpoint `{AI_API_URL}/admin/quota/model`

For example:

```
curl --location "$AI_API_URL/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  
    ]  
  }  
]
```

By default, tenant-level quota applies to all resource groups. If quota is requested for a specific resource-group, it'll appear as part of the response. To get a resource group level quota, include a resource group in your request. For example:

```
curl --location "${AI_API_URL}/admin/quota/model" \  
--header 'AI-Resource-Group: <Resource Group Id>' \  
--header "Authorization: Bearer $AUTH_TOKEN"
```

## Create a Rate Limit Update

You can request a rate limit update by sending a POST request to the endpoint `{${AI_API_URL}}/admin/quota/requests`. Include the details of your rate limit update in your request

### Tenant level Request

Populate the following request with your values:

```
curl --location "${AI_API_URL}/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.

For example:

```
curl --location "${AI_API_URL}/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>"
}'
```

## List Rate Limit Update Results

You can check the status of your rate limits by sending a GET request to the endpoint `{${AI_API_URL}}/admin/quota/requests`.

For example:

```
curl --location "${AI_API_URL}/admin/quota/requests" \
--header "Authorization: Bearer $AUTH_TOKEN"
```

For resource group level requests, include your resource group in the header. For example:

```
curl --location "${AI_API_URL}/admin/quota/requests" \
--header 'AI-Resource-Group: <Resource Group Id>' \
--header "Authorization: Bearer $AUTH_TOKEN"
```

To follow up on a request create a SNOW support ticket for component CA-ML-AIC. For more information, see [Support Process \[page 616\]](#).

# 8 Predictive AI

SAP AI Core lets you make data-driven decisions confidently and efficiently, tailored to address business challenges. It handles large volumes of data and offers scalable machine learning capabilities to automate tasks like triaging customer feedback or tickets and performing classification tasks. SAP AI Core includes preconfigured SAP solutions and supports open-source machine learning frameworks. It integrates with Argo Workflow and KServe, and can be embedded into other applications.

## [ML Operations \[page 450\]](#)

This section guides you through the end-to-end AI lifecycle of SAP AI Core.

## [Metrics \[page 555\]](#)

The AI API provides the ability to track metrics, and to customize or filter which metrics are reported.

## 8.1 ML Operations

This section guides you through the end-to-end AI lifecycle of SAP AI Core.

### [Connect Your Data \[page 450\]](#)

Use cloud storage with SAP AI Core to store AI assets such as datasets and model files. You use Artifacts in SAP AI Core to reference to your AI Assets.

### [Train Your Model \[page 460\]](#)

You execute a training workflow to train your AI learning model.

### [Use Your Model \[page 509\]](#)

You deploy your AI learning model to run inferences against it.

**Parent topic:** [Predictive AI \[page 450\]](#)

## Related Information

### [Metrics \[page 555\]](#)

## 8.1.1 Connect Your Data

Use cloud storage with SAP AI Core to store AI assets such as datasets and model files. You use Artifacts in SAP AI Core to reference to your AI Assets.

### [Manage Files \[page 451\]](#)

An artifact refers to data or a file that is produced or consumed by executions or deployments. They are managed through SAP AI Core and your connected object store.

#### [Manage Files Using the Dataset API \[page 455\]](#)

Where direct access to files in the object store is not possible or desirable (for example, in Content as a Service Scenarios, where the Service Consumers might not be the owners of the object store) you can upload, download, and delete files from the pre-registered object store using the SAP AI Core Dataset API.

**Parent topic:** [ML Operations \[page 450\]](#)

## Related Information

[Train Your Model \[page 460\]](#)

[Use Your Model \[page 509\]](#)

### 8.1.1.1 Manage Files

An artifact refers to data or a file that is produced or consumed by executions or deployments. They are managed through SAP AI Core and your connected object store.

The object store secret allows SAP AI Core to access your cloud storage and data without exposing your compromising your credentials.

[Create Files \[page 451\]](#)

[List Files \[page 454\]](#)

Use the *ML Operations* app to search for a dataset or artifact.

**Parent topic:** [Connect Your Data \[page 450\]](#)

## Related Information

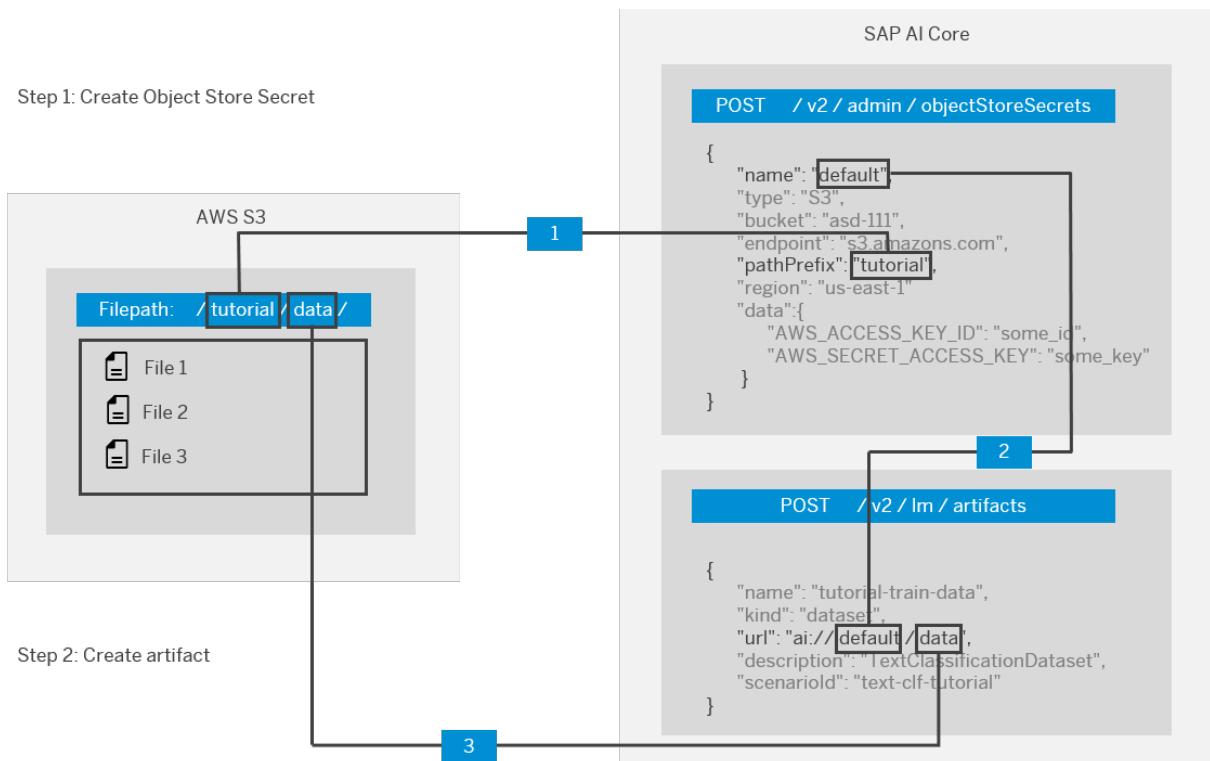
[Manage Files Using the Dataset API \[page 455\]](#)

### 8.1.1.1.1 Create Files

The object store secret lets SAP AI Core access your cloud storage and data, without exposing or compromising your credentials.

## ⚠ Restriction

The `objectStore_name`, `data_path` and `scenarioId` refer to pre-existing values. For the `objectStore_name` and `data_path` values, you must use the values that you used when registering the object storage, following the naming convention outlined in the diagram below. In example output code blocks, these values are represented by `ai://default/data`.



**Parent topic:** [Manage Files \[page 451\]](#)

## Related Information

[List Files \[page 454\]](#)

# Using a Third-Party API Platform

## Procedure

1. Create a new **POST** request using URL `{ {apiurl} }/v2/lm/artifacts`
2. Toggle the body tab, and enter the following JSON:

```
{
```

```
        "name": "name of artifact",
        "kind": "dataset",
        "url": "ai://<objectStore name>/<data path>",
        "description": "<description of artifact>",
        "scenarioId": "<scenarioID>"
    }
```

## Results

The response body contains the ID of your new artifact.

```
{
  "id": "3x4mpl3-651c-4f3e-8e1d-81a408041bc1",
  "message": "Artifact acknowledged",
  "url": "ai://default/data"
}
```

## Using curl

### Procedure

Run the following code:

```
curl --location --request POST "$API_URL/v2/lm/artifacts" \
--header "Authorization: Bearer $TOKEN" \
--header "Content-Type: application/json" \
--header "AI-Resource-Group: <Resource group>" \
--data-raw '{
  "name": "name of artifact",
  "kind": "dataset",
  "url": "ai://<objectStore name>/<data path>",
  "description": "<description of artifact>",
  "scenarioId": "<scenarioID>"
}'
```

## Results

The response body contains the ID of your new artifact.

```
{
  "id": "3x4mpl3-651c-4f3e-8e1d-81a408041bc1",
  "message": "Artifact acknowledged",
  "url": "ai://default/data"
}
```

## 8.1.1.2 List Files

Use the [ML Operations](#) app to search for a dataset or artifact.

**Parent topic:** [Manage Files \[page 451\]](#)

### Related Information

[Create Files \[page 451\]](#)

## Using Curl

### Procedure

Run the following code:

```
curl --request GET "$AI_API_URL/v2/lm/artifacts" --header "Authorization: Bearer $TOKEN" --header "AI-Resource-Group: $RESOURCE_GROUP"
```

### Results

#### ↔ Sample Code

```
{
  "count":3,
  "resources":[
    {
      "createdAt":"2021-02-09T08:08:12Z",
      "description":"",
      "executionId":"d44edae36c187cf6",
      "id":"3088b75f-5448-4c19-8055-392668a043ec",
      "kind":"model",
      "modifiedAt":"2021-02-09T08:08:12Z",
      "name":"pytf-model",
      "scenarioId":"ae0bd260-41ef-4162-81b0-861bd78a8516",
      "url":"ai://default/d44edae36c187cf6/pytf-model"
    },
    {
      "createdAt":"2021-02-09T07:56:37Z",
      "description":"",
      "executionId":"d44edae36c187cf6",
      "id":"38f7a46b-454d-4543-9457-b1eede5036f8",
      "kind":"model",
      "modifiedAt":"2021-02-09T07:56:37Z",
      "name":"churn-pickle",
      "scenarioId":"ae0bd260-41ef-4162-81b0-861bd78a8516",
      "url":"ai://default/d44edae36c187cf6/churn-pickle"
    }
  ]
}
```

```

        },
        {
            "createdAt": "2021-02-07T16:07:16Z",
            "description": "Churn and Text Classifier Dataset",
            "id": "b45265f2-9bc3-441a-a0e1-fac1438acb79",
            "kind": "dataset",
            "modifiedAt": "2021-02-07T16:07:16Z",
            "name": "pytf",
            "scenarioId": "84fe6957-1145-4183-b682-8f11ca56d060",
            "url": "ai://default/pytf/"
        }
    ]
}

```

## Using a Third-Party API Platform

### Procedure

1. Send a GET request to the endpoint `{apiurl}/v2/lm/artifacts`
2. On the *Authorization* tab, set the type to *Bearer Token*.
3. Set the token value to `{{{token}}}`.
4. On the *Header* tab, add the following entry:

Key	Value
AI-Resource-Group	<code>&lt;Name of your resourceGroup&gt;</code> (in the example, default is used)

5. Send the request.

### 8.1.1.2 Manage Files Using the Dataset API

Where direct access to files in the object store is not possible or desirable (for example, in Content as a Service Scenarios, where the Service Consumers might not be the owners of the object store) you can upload, download, and delete files from the pre-registered object store using the SAP AI Core Dataset API.

For full details on the Dataset API specification, see [SAP AI Core API documentation](#).

### Prerequisites

- You must have an object store secret defined in the target resource group.

#### ⚠ Restriction

Only S3 object stores can be used with the Dataset API.

### **⚠ Restriction**

Only `csv` type files can be uploaded with the Dataset API.

[Create Files \[page 456\]](#)

Where direct access to files in the object store is not possible or desirable (for example, in Content as a Service Scenarios, where the Service Consumers might not be the owners of the object store) you can upload, download, and delete files from the pre-registered object store using the SAP AI Core Dataset API.

[Download Files \[page 458\]](#)

[Delete Files \[page 459\]](#)

**Parent topic:** [Connect Your Data \[page 450\]](#)

## **Related Information**

[Manage Files \[page 451\]](#)

### **8.1.1.2.1 Create Files**

Where direct access to files in the object store is not possible or desirable (for example, in Content as a Service Scenarios, where the Service Consumers might not be the owners of the object store) you can upload, download, and delete files from the pre-registered object store using the SAP AI Core Dataset API.

## **Using Curl**

### **Prerequisites**

- You must have an object store secret defined in the target resource group.

### **⚠ Restriction**

Only S3 object stores can be used with the Dataset API.

### **⚠ Restriction**

Only `csv` type files can be uploaded with the Dataset API.

## Context

For full details on the Dataset API specification, see [SAP AI Core API documentation](#).

## Procedure

Run the following code:

```
curl --location --request PUT "$AI_API_URL/v2/lm/dataset/files/$SECRET_NAME/$FILE_PATH" \
--header "Authorization: Bearer $TOKEN" \
--header "Content-Type: text/csv" \
--header "AI-Resource-Group: $RESOURCE_GROUP" \
--data @$FILE_LOCATION
```

- If you use a service provider token, specify the resource group.  
If you use a service consumer token, the resource group information is contained in the token, and need not be specified.
- In the request body, submit the file as binary data.

# Using a Third-Party API Platform

## Prerequisites

- You must have an object store secret defined in the target resource group.

### ⚠ Restriction

Only S3 object stores can be used with the Dataset API.

### ⚠ Restriction

Only `csv` type files can be uploaded with the Dataset API.

## Context

For full details on the Dataset API specification, see [SAP AI Core API documentation](#).

## Procedure

1. Send a PUT request to the endpoint `{apiurl}/v2/lm/dataset/files/{secret name}/{full file path}`
2. In the header set the Content-Type as `<text/csv>`.
3. If you use a service provider token, specify the resource group. If you use a service consumer token, the resource group information is contained in the token, and need not be specified.
4. In the request body, submit the file as binary data.

## Results

Your file will be uploaded to the S3 storage bucket with the prefix specified in the object store secret, and the full file path specified in the upload request.

### ↔ Output Code

```
{  
  "message": "File default/test.csv created successfully.",  
  "url": "ai://default/test.csv"  
}
```

### 8.1.1.2.2 Download Files

## Using Curl

### Procedure

Run the following code:

```
curl --location --request GET "$AI_API_URL/v2/lm/dataset/files/$SECRET_NAME/  
$FILE_PATH" \  
  --header "Authorization: Bearer $TOKEN" \  
  --header "AI-Resource-Group: $RESOURCE_GROUP" \  
  --data @$FILE_LOCATION
```

## Using a Third-Party API Platform

### Procedure

Send a GET request to the endpoint {{apiurl}}/v2/lm/dataset/files/{{secret name}}/{{full file path}}

### 8.1.1.2.3 Delete Files

## Using Curl

### Procedure

Run the following code:

```
curl --location --request DELETE "$AI_API_URL/v2/lm/dataset/files/$SECRET_NAME/$FILE_PATH" \
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP"
```

## Using a Third-Party API Platform

### Procedure

Send a DELETE request to the endpoint {{apiurl}}/v2/lm/dataset/files/{{secret name}}/{{full file path}}

Ensure that the following headers are selected:

Key	Value
AI-Resource-Group	{{resource-group}}
Authorization	{{token}}

## 8.1.2 Train Your Model

You execute a training workflow to train your AI learning model.

SAP AI Core uses data as specified by your resource group. Models are specified in your workflow and are trained on a remote server. The resulting artifacts are stored to your object store.

The execution service includes:

- Batch jobs/pipelines via a state-of-the-art workflow engine.
- Parameterized templates for pipelines stored and managed at tenant level.
- Pipeline instances with specific parameters and resource isolation.
- Support for bringing own Docker registries and Docker images for running pipeline steps.
- Data extraction and validation modeled as pipeline steps.
- Bring your own hyperscaler-backed object store buckets.
- Dedicated object store buckets used as data storage. A **default** bucket should be provided by users per resource group.

### ⚠ Restriction

You must create an **object store secret** named **default** to store the training output artifact (for example, a model). If this default object store secret is missing, the training pipeline fails.

For **input training artifacts only**, you can create multiple object store secrets with different names as needed.

For more information, see [Register an Object Store Secret \[page 96\]](#).

The execution engine in SAP AI Core leverages the [Argo Workflows](#) open source project. It supports container-native workflows and pipelines modeled as direct acyclic graphs or steps. The Argo Workflows are used to ingest data, perform preprocessing and postprocessing, and train models and execute batch inference pipelines. SAP AI Core also leverages the parallel processing of steps in the form of a [DAG \(Directed Acyclic Graph\) structure](#) in workflow templates. For information about how using parallel nodes may affect your costs, see [Metering and Pricing for SAP AI Core \[page 9\]](#).

### ⓘ Note

Argo Workflow isn't optimized for time critical tasks. Each step must be scheduled onto a node in the cluster, and the cluster initialized. The time this takes depends on the load of the workflow controller and the node availability in the cluster. Therefore, it isn't recommended to use multistep Argo Workflows for time-critical tasks.

It's possible to train the same model multiple times, with varying parameters (for parameters compatible with the workflow, only) and evaluate them in SAP AI Launchpad.

#### [Choose an Instance \[page 462\]](#)

You can configure SAP AI Core to use different infrastructure instances for different tasks, based on demand. SAP AI Core provides several preconfigured infrastructure bundles called "resource plans" and "instance types" for this purpose.

#### [Workflow Templates \[page 464\]](#)

Here, you'll find a basic workflow example template. Feel free to adjust it to suit your workflow needs.

#### [List Scenarios \[page 469\]](#)

A scenario is an implementation of a specific AI use case within a user's tenant. It consists of a pre-defined set of AI capabilities in the form of executables and templates.

#### [List Executables \[page 472\]](#)

An executable is a reusable template that defines a workflow or pipeline for tasks such as training a machine learning model or creating a deployment. It contains placeholders for input artifacts (datasets or models) and parameters (custom key-pair values) that enable the template to be reused in different scenarios.

#### [Create Configurations \[page 478\]](#)

A configuration is a collection of parameters, artifact references (such as datasets or models), and environment settings that are used to instantiate and run an execution or deployment of an executable or template.

#### [List Configurations \[page 480\]](#)

#### [Using Artifact Signatures \[page 482\]](#)

Artifact signatures in the form of a hash can be added to output artifacts from executions.

#### [Start Training \[page 486\]](#)

#### [Stop Training Instances \[page 488\]](#)

#### [Delete Training Instances \[page 492\]](#)

#### [Efficiency Features \[page 496\]](#)

Discover features of the SAP AI Core runtime that improve efficiency and help manage resource consumption.

#### [Retrieve Execution Logs \[page 499\]](#)

Deployment and execution logs contain information about API processing and metrics.

#### [Training Schedules \[page 503\]](#)

**Parent topic:** [ML Operations \[page 450\]](#)

## Related Information

[Connect Your Data \[page 450\]](#)

[Use Your Model \[page 509\]](#)

## 8.1.2.1 Choose an Instance

You can configure SAP AI Core to use different infrastructure instances for different tasks, based on demand. SAP AI Core provides several preconfigured infrastructure bundles called “resource plans” and “instance types” for this purpose.

### Context

You must choose at least one instance. If you select a resource plan, an instance type isn't required and vice versa.

Resource plans and instance types are used to select instances in workflow and serving templates. Different steps of a workflow can have different instances assigned.

In general, if your workload needs GPU acceleration, use one of the GPU-enabled instances. Otherwise, choose a plan based on the anticipated CPU and memory need of your workloads.

Within SAP AI Core, the instances are selected via the `ai.sap.com/resourcePlan` and `ai.sap.com/instanceType` labels at pod level. It maps the selected instance and takes a string value, which is the ID of the instance.

There are limits to the default disk storage size for all these nodes. Datasets that are loaded to the nodes consume disk space. If you have large data sets (larger than 30 GB), or have large models, you may have to increase the disk size. To do so, use the persistent volume claim in Argo Workflows to specify the required disk size (see [Volumes](#) ).

### Instance Types

#### ⚠ Restriction

Instance types are only available as part of the “extended” service plan. For more information on instances that can be used with the “standard” service plan.

Within SAP AI Core, the instances are selected via the `ai.sap.com/instanceType` label in your workflow templates. It maps the selected resource and takes a string value, which is the ID of the instance

For information about available instance types and their IDs, see SAP Note [3660109](#), or send a GET request to the following endpoint:

```
{{apiurl}}/v2/admin/resources/instanceType
```

For example:

#### ↔ Sample Code

```
curl GET "$AI_API_URL/v2/admin/resources/instanceType" \
--header "Authorization: Bearer $TOKEN" \
```

## Resource Plans

Within SAP AI Core, the instances are selected via the `ai.sap.com/resourcePlan` label in your workflow templates. It maps the selected resource and takes a string value, which is the ID of the instance.

Resource Plan Specifications for AWS

Resource Plan IDs	GPUs	CPU Cores	Memory GBs	Code to Allocate instances in Workflow Templates
Starter	-	1	3	<code>ai.sap.com/resourcePlan:starter</code>
Basic	-	3	11	<code>ai.sap.com/resourcePlan:basic</code>
Basic-8x	-	31	116	<code>ai.sap.com/resourcePlan:basic.8x</code>
Train-L	1 V100	7	55	<code>ai.sap.com/resourcePlan:train.l</code>
Infer-S	1 T4	3	10	<code>ai.sap.com/resourcePlan:infer.s</code>
Infer-M	1 T4	7	26	<code>ai.sap.com/resourcePlan:infer.m</code>
Infer-L	1 T4	15	58	<code>ai.sap.com/resourcePlan:infer.l</code>

### ⓘ Note

For the free service plan, only the Starter resource plan is available. Specifying other plans results in an error. For the standard service plan, all resource plans are available.

[Parent topic: Train Your Model \[page 460\]](#)

## Related Information

[Workflow Templates \[page 464\]](#)

[List Scenarios \[page 469\]](#)

[List Executables \[page 472\]](#)  
[Create Configurations \[page 478\]](#)  
[List Configurations \[page 480\]](#)  
[Using Artifact Signatures \[page 482\]](#)  
[Start Training \[page 486\]](#)  
[Stop Training Instances \[page 488\]](#)  
[Delete Training Instances \[page 492\]](#)  
[Efficiency Features \[page 496\]](#)  
[Retrieve Execution Logs \[page 499\]](#)  
[Training Schedules \[page 503\]](#)

## Service Usage Reporting

Usage consumption of services is reported in the SAP BTP cockpit on the [Overview](#) page for your global account and on the [Overview](#) and [Usage Analytics](#) pages of your subaccount. The usage report lists usage in billable measures and non-billable measures. Your final monthly bill is based on the billable measure only. Non-billable measures are displayed for reporting purposes only.

### 8.1.2.2 Workflow Templates

Here, you'll find a basic workflow example template. Feel free to adjust it to suit your workflow needs.

Workflow templates help manage your training pipelines at the main tenant level. You can store these templates in your git repository and version them as needed. SAP AI Core executes workflows using the Argo Workflows open source project. Argo Workflows is an open-source, container-native workflow engine that orchestrates parallel jobs on Kubernetes. It's implemented as a Kubernetes CRD (Custom Resource Definition).

Workflow templates act as executables. To map them, you need certain attributes in your template's metadata section. The AI API uses the annotations and labels in the template to find scenarios and executables.

Workflow templates are built on the Argo Workflows engine and are defined as `WorkflowTemplates`. These are your cluster's workflow definitions. For more information about `WorkflowTemplates` and sample workflows, see the Argo documentation.

In SAP AI Core, Argo Workflows are used to:

- Train models
- Ingest data
- Preprocess and postprocess data
- Execute batch inference pipelines

Workflows are executed in batch mode. A workflow can produce multiple output artifacts, but only an output artifact with a `globalName` is considered to be the final output artifact of the workflow.

## **⚠ Restriction**

Output artifacts can only use the **default** object store.

For the model training code, SAP AI Core is language-agnostic. However, you need to specify the relevant programming language in the workflow parameters. If you're importing any packages, list them in a separate file named `requirements.txt` and store it in the same directory.

## **⚠ Restriction**

The maximum number of workflow templates is limited at tenant level to 50. If you reach this limit, you receive an error message. To free up space, delete some workflow templates. Alternatively, raise a ticket to increase your quota.

To get started, copy the generic workflow template below and add your own values as required. You can use any text editor with a YAML plugin to create your templates. Workflows support the following parameters:

Workflow Template Parameters

Type	Parameter	Description
name (mandatory)	-	The executable ID. The executable ID must be unique among all executables available within the SAP AI Core main tenant.
labels (mandatory)	<code>ai.sap.com/resourcePlan</code>	You must specify the chosen <code>resourcePlan</code> . The value is the string value of the selected resource plan. For more information, see <a href="#">Choose an Instance [page 462]</a> .
annotations	<code>scenarios.ai.sap.com/description</code> (optional)	A description of the scenario to which this executable belongs.
	<code>scenarios.ai.sap.com/name</code> (mandatory)	The scenario name, which is necessary for the AI API to discover a scenario.
	<code>executables.ai.sap.com/description</code> (optional)	The description of an executable.
	<code>executables.ai.sap.com/name</code> (mandatory)	The name of the executable.
	<code>artifacts.ai.sap.com/&lt;argo_artifact_name&gt;.suffix</code> x	A suffix added to the file or folder of the output artifact on the object store.
	<code>artifacts.ai.sap.com/&lt;argo_artifact_name&gt;.kind</code>	The type of output artifact (for example, a dataset or model).
	<code>artifacts.ai.sap.com/&lt;argo_artifact_name&gt;.name</code>	The name under which the output artifact is registered in the AI API.

Type	Parameter	Description
	<pre>artifacts.ai.sap.com/ &lt;argo_artifact_name&gt;.description (optional)</pre> <pre>artifacts.ai.sap.com/ &lt;argo_artifact_name&gt;.label s:   {"ext.ai.sap.com/ customkey1": "customvalue1" , "ext.ai.sap.com/ customkey2": "customvalue2" } (optional)</pre>	You can add further metadata for an artifact using these annotations.
labels	<pre>scenarios.ai.sap.com/id (mandatory)</pre>	The scenario ID to which the workflow template belongs.
	<pre>ai.sap.com/version (mandatory)</pre>	The compatibility version of this executable. You can use the compatibility versions to offer executables in different variants for different AI service consumers.

### ⓘ Note

In the artifact-related parameters above, <argo\_artifact\_name> refers to the globalName of an output artifact.

## Generic Workflow Template

```
apiVersion: argoproj.io/v1alpha1
kind: WorkflowTemplate
metadata:
  name: text-clf-train-tutorial
  annotations:
    scenarios.ai.sap.com/description: "SAP developers tutorial scenario"
    scenarios.ai.sap.com/name: "text-clf-tutorial-scenario"
    executables.ai.sap.com/description: "Text classification Scikit training executable"
    executables.ai.sap.com/name: "text-clf-train-tutorial-exec"
    artifacts.ai.sap.com/text-data.kind: "dataset"
    artifacts.ai.sap.com/text-model-tutorial.kind: "model"
    artifacts.ai.sap.com/text-model-tutorial.description: "artifact description"
    artifacts.ai.sap.com/text-model-tutorial.labels: |
      {"ext.ai.sap.com/customkey1": "customvalue1", "ext.ai.sap.com/
customkey2": "customvalue2"}
    labels:
      scenarios.ai.sap.com/id: "text-clf-tutorial"
      executables.ai.sap.com/id: "text-clf-train-tutorial"
      ai.sap.com/version: "1.0.0"
spec:
  imagePullSecrets:
    - name: <name of your Docker registry secret>
```

```

entrypoint: text-clf-sk-training
arguments:
  parameters: # placeholder for string like inputs
    - name: DEPTH # identifier local to this workflow
      description: description of the parameter
      default: test
templates:
  - name: text-clf-sk-training
    metadata:
      labels:
        ai.sap.com/resourcePlan: starter
    inputs:
      artifacts:
        - name: text-data
          path: /app/data/
    outputs:
      artifacts:
        - name: text-model-tutorial
          path: /app/model
          globalName: text-model-tutorial
          archive:
            none: {}
    container:
      image: "<DOCKER IMAGE URL GOES HERE>"
      imagePullPolicy: Always
      command: ["/bin/sh", "-c"]
      args:
        - >
          set -e && echo "---Start Training---" && python /app/src/
train_scikit.py && ls -lR /app/model && echo "----End Training---"

```

### → Tip

It is recommended that you specify a value for `activeDeadlineSeconds` in the workflow template. `activeDeadlineSeconds` prevents the workflow from running indefinitely and incurring high costs, for example if a linux process gets stuck. The `activeDeadlineSeconds` parameter is specified in the `spec` section of the workflow template, and defines the maximum duration in seconds that the workflow can run before it is stopped. For more information, see [Argo Workflow Timeouts](#).

### ⓘ Note

For every container in the template, the `command: ["/bin/sh", "-c"]` field is mandatory. The contents of the argument can be amended, but must not be empty. The `CMD` and `ENDPOINT` specified in the Dockerfile of a container are ignored.

User ID and group ID 65534 is required to run the Docker image. This user has permission to access the files while the application is running. You can check and change the permissions by using the `chown` and `chmod` commands. For more information about security-related restrictions, see [Kubernetes Security](#) [page 589].

### → Tip

To make sure that the container is working as expected before submitting it to SAP AI Core, run `docker run -it --user 65534:65543 <docker-image>` locally.

### ⓘ Note

The `archive: none: {}` option in the `outputs artifacts` sections disables automatic archiving for the artifact. If archiving is enabled, the output artifacts are archived to a `tar-gzip` file before they are uploaded to the object store. If archiving is disabled through `archive: {}`, the artifact will be

uploaded to the object store in its current format. If the output artifact points to a directory, the directory will be uploaded “as is” to the object store. However, object stores in SAP HANA Cloud, data lake do not support this. In this case, remove `archive: none: {}` and archive the directory to a single `tar-gzip` file before it is uploaded to the object store.

### Note

When multiple containers are defined in a single WorkflowTemplate, for example when using Sidecar or Container Set, one of the containers must be named `main`. To fetch logs of another container in the same template using the a GET request to the endpoint `/logs`, the name of the container must have the ``readable-`` prefix. The execution can still run without the ``readable-`` prefix, but logs will be inaccessible through the endpoint.. For more information, see [Argo Sidecar](#) and [Argo Container Set](#).

## Sync an Application Manually

Applications sync with your GitHub repository automatically at intervals of ~3 minutes. Use the endpoint below to manually request a sync:`{apiurl}/admin/applications/{appName}/refresh`

**Parent topic:** [Train Your Model \[page 460\]](#)

## Related Information

[Choose an Instance \[page 462\]](#)  
[List Scenarios \[page 469\]](#)  
[List Executables \[page 472\]](#)  
[Create Configurations \[page 478\]](#)  
[List Configurations \[page 480\]](#)  
[Using Artifact Signatures \[page 482\]](#)  
[Start Training \[page 486\]](#)  
[Stop Training Instances \[page 488\]](#)  
[Delete Training Instances \[page 492\]](#)  
[Efficiency Features \[page 496\]](#)  
[Retrieve Execution Logs \[page 499\]](#)  
[Training Schedules \[page 503\]](#)

## Stop or Delete Multiple Training Instances

The feature is set to false by default. To enable bulk PATCH operations, your template must contain the following snippet, with the relevant values set to `true`.

```
Meta:  
  "bulkUpdates": {  
    "executions": false,  
    "deployments": false  
  }
```

## Related Information

[Argo Workflows](#) ↗

- [Register Your Docker Registry Secret \[page 107\]](#)
- [Stop Multiple Training Instances \[page 490\]](#)
- [Delete Multiple Training Instances \[page 494\]](#)

### 8.1.2.3 List Scenarios

A scenario is an implementation of a specific AI use case within a user's tenant. It consists of a pre-defined set of AI capabilities in the form of executables and templates.

**Parent topic:** [Train Your Model \[page 460\]](#)

## Related Information

- [Choose an Instance \[page 462\]](#)
- [Workflow Templates \[page 464\]](#)
- [List Executables \[page 472\]](#)
- [Create Configurations \[page 478\]](#)
- [List Configurations \[page 480\]](#)
- [Using Artifact Signatures \[page 482\]](#)
- [Start Training \[page 486\]](#)
- [Stop Training Instances \[page 488\]](#)
- [Delete Training Instances \[page 492\]](#)
- [Efficiency Features \[page 496\]](#)
- [Retrieve Execution Logs \[page 499\]](#)
- [Training Schedules \[page 503\]](#)

# List Scenarios Using Curl

## Procedure

Run the following code:

```
curl --request GET "$AI_API_URL/v2/lm/scenarios" --header "Authorization: Bearer $TOKEN" --header "AI-Resource-Group: $RESOURCE_GROUP"
```

## Results

### ↔ Output Code

```
{
  "count":2,
  "resources":[
    {
      "createdAt":"2021-02-03T18:38:32+00:00",
      "description":"churn and text class scenario desc",
      "id":"84fe6957-1145-4183-b682-8f11ca56d060",
      "labels":[
        ],
      "modifiedAt":"2021-02-04T11:14:02+00:00",
      "name":"churntextclassscenname"
    },
    {
      "createdAt":"2021-02-04T14:11:02+00:00",
      "description":"churn and text class scenario desc",
      "id":"ae0bd260-41ef-4162-81b0-861bd78a8516",
      "labels":[
        ],
      "modifiedAt":"2021-02-09T07:35:03+00:00",
      "name":"churntextclassscenname"
    }
  ]
}
```

### ⓘ Note

Only scenarios that have a defined training executable can be created. When a new scenario ID is specified in the workflow template for a training executable, a new scenario will be created along with the training executable.

For now, a new scenario with only a deployment executable cannot be created. A possible work-around is to create a scenario with a dummy training executable and then use the same scenario ID in the serving template.

## Get Scenario Versions Using Curl

### Procedure

Run the following code:

```
curl --location --request GET '$API_URL/v2/lm/scenarios/$SCENARIO_ID/versions' \
```

## List Scenarios Using a Third-Party API Platform

### Procedure

1. Send GET request to the endpoint {{apiurl}}/v2/lm/scenarios
2. On the *Authorization* tab, set the type to *Bearer Token*.
3. Set the token value to {{token}}.
4. On the *Header* tab, add the following entry:

Key	Value
AI-Resource-Group	<Name of your resourceGroup>

5. Send the request.

#### ⓘ Note

Only scenarios that have a defined training executable can be created. When a new scenario ID is specified in the workflow template for a training executable, a new scenario will be created along with the training executable.

For now, a new scenario with only a deployment executable cannot be created. A possible work-around is to create a scenario with a dummy training executable and then use the same scenario ID in the serving template.

## Get Scenario Versions Using a Third-Party API Platform

### Procedure

Create a new GET request and enter the URL {{apiurl}}/v2/lm/scenarios/{{scenarioid}}/versions

## 8.1.2.4 List Executables

An executable is a reusable template that defines a workflow or pipeline for tasks such as training a machine learning model or creating a deployment. It contains placeholders for input artifacts (datasets or models) and parameters (custom key-pair values) that enable the template to be reused in different scenarios.

**Parent topic:** [Train Your Model \[page 460\]](#)

### Related Information

[Choose an Instance \[page 462\]](#)

[Workflow Templates \[page 464\]](#)

[List Scenarios \[page 469\]](#)

[Create Configurations \[page 478\]](#)

[List Configurations \[page 480\]](#)

[Using Artifact Signatures \[page 482\]](#)

[Start Training \[page 486\]](#)

[Stop Training Instances \[page 488\]](#)

[Delete Training Instances \[page 492\]](#)

[Efficiency Features \[page 496\]](#)

[Retrieve Execution Logs \[page 499\]](#)

[Training Schedules \[page 503\]](#)

## List Executables Using Curl

### Procedure

Run the following code:

```
curl --request GET "$AI_API_URL/v2/lm/scenarios" --header "Authorization: Bearer $TOKEN" --header "AI-Resource-Group: $RESOURCE_GROUP"
```

### Results

#### ↔ Output Code

```
{
  "count": 4,
  "resources": [
    {
      "createdAt": "2021-02-04T13:11:01+00:00",
```

```

"deployable":true,
"description":"churn n text class serving executable desc",
"id":"pytf-serving",
"inputArtifacts":[
    {
        "name":"model_uri"
    }
],
"labels":[
],
"modifiedAt":"2021-02-04T13:11:01+00:00",
"name":"churntextclassexecname",
"parameters":[
    {
        "name":"modelName",
        "type":"string",
        "default": "value",
        "description": "description of the parameter"
    }
],
"scenarioId":"ae0bd260-41ef-4162-81b0-861bd78a8516",
"versionId":"0.0.1"
},
{
    "createdAt":"2021-02-09T07:35:02+00:00",
    "deployable":true,
    "description":"churn n text class serving executable desc",
    "id":"pytf-serving-tracking",
    "inputArtifacts":[
        {
            "name":"textmodel",
            "kind": "model",
            "description": "artifact description",
            "labels": [
                {
                    "key": "ext.ai.sap.com/customkey1",
                    "value": "customvalue1"
                },
                {
                    "key": "ext.ai.sap.com/customkey2",
                    "value": "customvalue2"
                }
            ]
        }
    ],
    "labels":[
],
    "modifiedAt":"2021-02-09T07:35:02+00:00",
    "name":"churntextclassexecname",
    "parameters":[
        {
            "name":"modelName",
            "type":"string"
        }
    ],
    "scenarioId":"ae0bd260-41ef-4162-81b0-861bd78a8516",
    "versionId":"0.0.1"
},
{
    "createdAt":"2021-02-09T07:35:03+00:00",
    "deployable":false,
    "description":"churn and text class executable desc",
    "id":"pytf-training-tracking",
    "inputArtifacts":[
        {
            "name":"churn-data"
        }
    ]
}

```

```

        },
        {
            "name": "textclass-data"
        }
    ],
    "labels": [
        ],
        "modifiedAt": "2021-02-09T07:35:03+00:00",
        "name": "churnnntextclassexecutablename",
        "outputArtifacts": [
            {
                "name": "churn-pickle",
                "kind": "model",
                "description": "artifact description",
                "labels": [
                    {
                        "key": "ext.ai.sap.com/customkey1",
                        "value": "customvalue1"
                    },
                    {
                        "key": "ext.ai.sap.com/customkey2",
                        "value": "customvalue2"
                    }
                ]
            },
            {
                "name": "pytf-model"
            }
        ],
        "parameters": [
            {
                "name": "train-epoch",
                "type": "string"
            }
        ],
        "scenarioId": "ae0bd260-41ef-4162-81b0-861bd78a8516",
        "versionId": "0.0.1"
    },
    {
        "createdAt": "2021-02-04T14:11:02+00:00",
        "deployable": false,
        "description": "churn and text class executable desc",
        "id": "test-training",
        "inputArtifacts": [
            {
                "name": "churn-data"
            },
            {
                "name": "textclass-data"
            }
        ],
        "labels": [
        ],
        "modifiedAt": "2021-02-04T14:11:02+00:00",
        "name": "churnnntextclassexecutablename",
        "outputArtifacts": [
            {
                "name": "churn-pickle"
            },
            {
                "name": "pytf-model"
            }
        ],
        "parameters": [
            {
                "name": "train-epoch",
            }
        ]
    }
]

```

```

        "type": "string"
    }
],
"scenarioId": "ae0bd260-41ef-4162-81b0-861bd78a8516",
"versionId": "0.0.1"
}
}
}
```

### ⓘ Note

The <modifiedAt> field denotes the timestamp of the latest successful sync. The output 1970-01-01T00:00:00+00:00 indicates an error.

## Get Executable Details Using Curl

### Procedure

Run the following code:

```
curl --request GET "{{apiurl}}/v2/lm/scenarios/{{scenarioid}}/executables"
--header "Authorization: Bearer $TOKEN" --header "AI-Resource-Group:
$RESOURCE_GROUP"
```

### Results

#### ↗ Output Code

```
{
  "createdAt": "2021-02-04T14:11:02+00:00",
  "deployable": false,
  "description": "churn and text class executable desc",
  "id": "test-training",
  "inputArtifacts": [
    {
      "name": "churn-data"
    },
    {
      "name": "textclass-data"
    }
  ],
  "labels": [
  ],
  "modifiedAt": "2021-02-04T14:11:02+00:00",
  "name": "churnntextclassexecutablename",
  "outputArtifacts": [
    {
      "name": "churn-pickle"
    },
    {
      "name": "pytf-model"
    }
  ]
}
```

```

        }
    ],
    "parameters": [
        {
            "name": "train-epoch",
            "type": "string"
        }
    ],
    "scenarioId": "ae0bd260-41ef-4162-81b0-861bd78a8516",
    "versionId": "0.0.1"
}

```

### ⓘ Note

The `<modifiedAt>` field denotes the timestamp of the latest successful sync. The output `1970-01-01T00:00:00+00:00` indicates an error.

## List Executables Using a Third-Party API Platform

### Procedure

1. Add your scenario ID as the value for the `scenarioId` environment variable.
2. Send a GET request to the endpoint `{apiurl}/v2/lm/scenarios/{scenarioId}/executables`
3. On the *Authorization* tab, set the type to *Bearer Token*.
4. Set the token value to `{token}`.
5. On the *Header* tab, add the following entry:

Key	Value
AI-Resource-Group	<code>&lt;Name of your resourceGroup&gt;</code> default is used

6. Send the request.

### ⇨ Output Code

```

{
  "count": 3,
  "resources": [
    {
      "createdAt": "2021-10-07T20:07:18+00:00",
      "deployable": true,
      "description": "Inference executable for text classification with Scikit-learn",
      "id": "text-clf-infer-tutorial",
      "input artifacts": [
        ...
      ]
    }
  ]
}

```

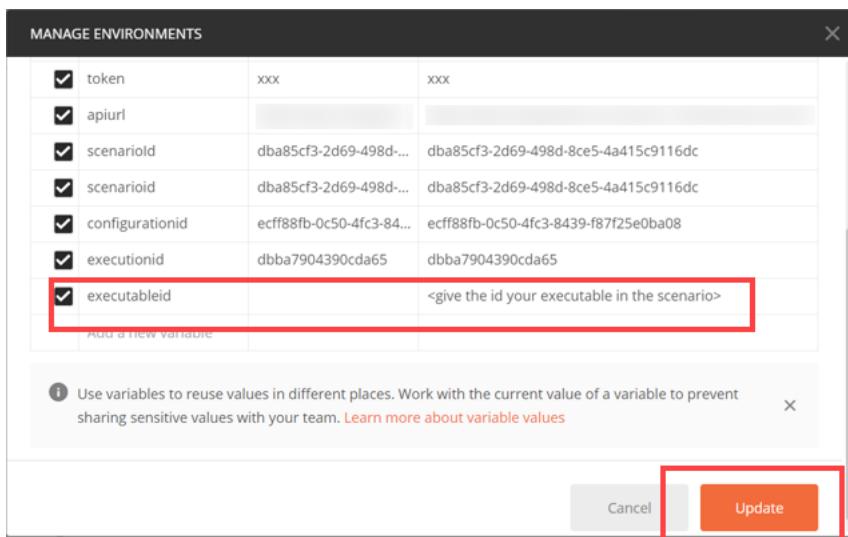
### ⓘ Note

The `<modifiedAt>` field denotes the timestamp of the latest successful sync. The output `1970-01-01T00:00:00+00:00` indicates an error.

# Get Executable Details Using a Third-Party API Platform

## Procedure

1. Add the environment variable executableid and as its value, enter the ID of the executable.



2. Send a GET request to the endpoint `{apiurl}/v2/lm/scenarios/{scenarioId}/executables/{executableId}`
3. On the *Authorization* tab, set the type to *Bearer Token*.
4. Set the token value to `{token}`.

The screenshot shows the 'Authorization' tab in Postman. The 'Type' dropdown is set to 'Bearer Token' and the 'Token' field contains '{token}'. A note about sensitive data is visible.

5. On the *Header* tab, add the following entry:

Key	Value
AI-Resource-Group	<Name of your resourceGroup> (in the example, default is used)

6. Send the request.

## 8.1.2.5 Create Configurations

A configuration is a collection of parameters, artifact references (such as datasets or models), and environment settings that are used to instantiate and run an execution or deployment of an executable or template.

**Parent topic:** [Train Your Model \[page 460\]](#)

### Related Information

[Choose an Instance \[page 462\]](#)

[Workflow Templates \[page 464\]](#)

[List Scenarios \[page 469\]](#)

[List Executables \[page 472\]](#)

[List Configurations \[page 480\]](#)

[Using Artifact Signatures \[page 482\]](#)

[Start Training \[page 486\]](#)

[Stop Training Instances \[page 488\]](#)

[Delete Training Instances \[page 492\]](#)

[Efficiency Features \[page 496\]](#)

[Retrieve Execution Logs \[page 499\]](#)

[Training Schedules \[page 503\]](#)

## Using Curl

### Procedure

Run the following code:

```
curl --request POST "$AI_API_URL/v2/lm/configurations" --header "Authorization: Bearer $TOKEN" --header "AI-Resource-Group: $RESOURCE_GROUP" --header "Content-Type: application/json" \
-d '{
    "name": "dummy-configuration",
    "executableId": """$EXECUTABLE""",
    "scenarioId": """$SCENARIO""",
    "parameterBindings": [
        {
            "key": "parameter_name_in_template",
            "value": "some_value"
        }
    ],
    "inputArtifactBindings": [
        {
            "key": "input_artifact_name_in_template",
            "artifactId": "a4d62a76-52aa-44cf-a789-743246d6d55b"
        }
    ]
}'
```

```
    ]  
}'
```

#### ↔ Output Code

```
{  
  "id": "f5bf305f-7c3f-4882-9f6b-8b95e3687b9b",  
  "message": "Configuration created"  
}
```

## Using a Third-Party API Platform

### Procedure

1. Send a POST request to the endpoint `{apiurl}/v2/lm/configurations`
2. On the *Authorization* tab, set the type to *Bearer Token*.
3. Set the token value to `{token}`.
4. On the *Header* tab, add the following entries:

KEY	VALUE
AI-Resource-Group	<Name of your resourceGroup>
Content-Type	<code>application/json</code>

5. On the *Body* tab, select the *raw* radio button and add the request body as given below:

```
{  
  "name": "configuration-name",  
  "executableId": "<executable ID>",  
  "scenarioId": "<scenario ID>",  
  "parameterBindings": [  
    {  
      "key": "<parameter name>",  
      "value": "<value>"  
    }  
  ],  
  "inputArtifactBindings": [  
    {  
      "key": "<artifact name>",  
      "artifactId": "<artifact ID>"  
    }  
  ]  
}
```

#### ↔ Output Code

```
...  
": [  
{  
  "key": "churn-data",  
  "artifactId": "896e9230-9219-47fa-89c7-c47db7d45d6a"
```

```
}
```

```
]
```

- Send the request.

### 8.1.2.6 List Configurations

**Parent topic:** Train Your Model [page 460]

#### Related Information

[Choose an Instance \[page 462\]](#)  
[Workflow Templates \[page 464\]](#)  
[List Scenarios \[page 469\]](#)  
[List Executables \[page 472\]](#)  
[Create Configurations \[page 478\]](#)  
[Using Artifact Signatures \[page 482\]](#)  
[Start Training \[page 486\]](#)  
[Stop Training Instances \[page 488\]](#)  
[Delete Training Instances \[page 492\]](#)  
[Efficiency Features \[page 496\]](#)  
[Retrieve Execution Logs \[page 499\]](#)  
[Training Schedules \[page 503\]](#)

## Using Curl

### Procedure

Run the following code:

```
curl --request GET "$AI_API_URL/v2/lm/configurations" --header "Authorization: Bearer $TOKEN" --header "AI-Resource-Group: $RESOURCE_GROUP"
```

#### Output Code

```
{
  "count": 2,
  "resources": [
    {
      "createdAt": "2021-02-04T11:50:45Z",
      "executableId": "pytf-training",
      "id": "1e6c2a5f-eabe-49a2-88e7-887145a2ef88",
```

```

    "inputArtifactBindings": [
        {
            "artifactId": "521f7f17-876e-4369-9162-09748b56d27a",
            "key": "churn-data"
        },
        {
            "artifactId": "521f7f17-876e-4369-9162-09748b56d27a",
            "key": "textclass-data"
        }
    ],
    "name": "pytf-demo-config1",
    "parameterBindings": [
        {
            "key": "train-epoch",
            "value": "100"
        }
    ],
    "scenarioId": "84fe6957-1145-4183-b682-8f11ca56d060"
},
{
    "createdAt": "2021-02-04T11:59:22Z",
    "executableId": "pytf-training",
    "id": "42147d0d-5e3a-424d-a3a1-545b842379c5",
    "inputArtifactBindings": [
        {
            "artifactId": "521f7f17-876e-4369-9162-09748b56d27a",
            "key": "churn-data"
        },
        {
            "artifactId": "521f7f17-876e-4369-9162-09748b56d27a",
            "key": "textclass-data"
        }
    ],
    "name": "pytf-demo-config2",
    "parameterBindings": [
        {
            "key": "train-epoch",
            "value": "1"
        }
    ],
    "scenarioId": "84fe6957-1145-4183-b682-8f11ca56d060"
}
]
}

```

## Using a Third-Party API Platform

### List Configurations

#### Procedure

1. Send a GET request to the endpoint {{apiurl}}/v2/lm/configurations
2. On the *Authorization* tab, set the type to *Bearer Token*.

3. Set the token value to `{ {token} }`.
4. On the [Header](#) tab, add the following entry:

KEY	VALUE
AI-Resource-Group	<Name of your resourceGroup>

5. Send the request.

```
{
  "count": 45,
  "resources": [
    {
      "createdAt": "2021-11-22T06:51:03Z",
      "executableId": "text-clf-train-tutorial-metrics",
    ...
  ]}
```

### 8.1.2.7 Using Artifact Signatures

Artifact signatures in the form of a hash can be added to output artifacts from executions.

#### Providing Signatures for Output Artifacts

A hash is an identifier that is generated using the file contents. Hashes can therefore be used to compare two files and verify the integrity of a file. SAP AI Core doesn't create or verify the signatures, but stores them and makes them available. Artifact signatures are created and stored with the artifact.

To provide a signature for an output artifact, specify the `AICORE_ARTIFACT_SIGNATURES` output parameter in the workflow template. For more information, see [Argo Workflows Output Parameters](#). This parameter specifies a path to a file that will contain the signatures as key-value-pairs in JSON format.

```
apiVersion: argoproj.io/v1alpha1
kind: WorkflowTemplate
metadata:
  name: signature-example
  annotations:
    scenarios.ai.sap.com/description: "Example for signature feature"
    scenarios.ai.sap.com/name: "signature-example-scenario"
    executables.ai.sap.com/description: "Output artifact example for signature feature"
    executables.ai.sap.com/name: "signature-example-output-artifacts"
  labels:
    scenarios.ai.sap.com/id: "signature-example-scenario"
    executables.ai.sap.com/id: "signature-example-out-art"
    ai.sap.com/version: "1.0.0"
spec:
  entrypoint: output-artifact-signatures-example
  templates:
    - name: output-artifact-signatures-example
      outputs:
        # specify an output artifact named "dummy-model"
        artifacts:
```

```

- name: dummy-model
  path: /tmp/model
  globalName: dummy-model
# specify an output parameter with name 'AICORE_ARTIFACT_SIGNATURES'
# this parameter should point to a file in valid JSON format, containing
an object with key-value pairs, where each key is
# the globalName of an output artifact and the value is its signature
parameters:
- name: AICORE_ARTIFACT_SIGNATURES # the parameter name has to be
exactly 'AICORE_ARTIFACT_SIGNATURES'
  valueFrom:
    path: /tmp/signatures.json # the value of the path can be set
arbitrary
container:
  image: python:3.11-alpine
  imagePullPolicy: Always
  command: ["python", "-c"]
args:
- |
  import hashlib, json
  model = "a dummy string serving as output artifact"
  model_signature =
hashlib.sha256(model.encode("utf-8")).hexdigest() # create a hash of the dummy
model
  signatures = {
    # the key is the globalName of the output artifact, the value
is its signature
    "dummy-model": model_signature
  }
  # write the dummy model to disk
  with open("/tmp/model", "w") as f:
    f.write(model)
  # write the model signatures dictionary to the path specified in
the AICORE_ARTIFACT_SIGNATURES output parameter
  with open("/tmp/signatures.json", "w") as f:
    json.dump(signatures, f)
  print("success")

```

The example template can be used with multiple output artifacts. For each artifact, you add another entry in the signatures object.

The template produces the output artifact `dummy-model`, with its name-signature key-value pair provided in the `AICORE_ARTIFACT_SIGNATURES` output parameter. The key is the global name of the output artifact, and the value is the signature. For example:

```
{
  "name-of-model1": "signature-of-model1"
}
```

These signatures can then be consumed in another execution.

### *ⓘ Note*

Signatures must be generated as part of the training workflow, and can't be added retrospectively.

### *ⓘ Note*

When you query the execution via the `GET {{apiurl}}/v2/lm/executions/{{executionid}}` endpoint you won't see the signatures of the output artifacts, as they're kept internally. However, you'll still be able to use them in other executions or deployments.

## Multi-Step Workflows

For multistep workflow templates, the `AICORE_ARTIFACT_SIGNATURES` output parameter needs to be supplied in each step that requires an output artifact and signature to be created. See the following example:

```
apiVersion: argoproj.io/v1alpha1
kind: WorkflowTemplate
metadata:
  name: signature-example-2
  annotations:
    scenarios.ai.sap.com/description: "Example for signature feature"
    scenarios.ai.sap.com/name: "signature-example-scenario"
    executables.ai.sap.com/description: "Output artifact example for multi
signature feature"
    executables.ai.sap.com/name: "signature-example-multi-output-artifacts"
  labels:
    scenarios.ai.sap.com/id: "signature-example-scenario"
    executables.ai.sap.com/id: "signature-example-multi-out-art"
    ai.sap.com/version: "1.0.0"
spec:
  entrypoint: steps-example
  templates:
    # a template with two steps, each having an output artifact
    - name: steps-example
      steps:
        - - output-artifact-signatures-step1
        - - output-artifact-signatures-step2
      name: output-artifact-signatures-step1
      outputs:
        # specify an output artifact named "dummy-model1"
        artifacts:
          - name: dummy-model1
            path: /tmp/model
            globalName: dummy-model1
        # the AICORE_ARTIFACT_SIGNATURES parameter can only hold artifact
signatures of the current step,
        # e.g. it could not hold the signature of "dummy-model2" below, because
that is created in a different workflow step
        parameters:
          - name: AICORE_ARTIFACT_SIGNATURES
            valueFrom:
              path: /tmp/signatures.json
      container:
        image: python:3.11-alpine
        imagePullPolicy: Always
        command: ["python", "-c"]
        args:
          - >
            # ... skipped, as it will be the same as in the previous example
      - name: output-artifact-signatures-step2
      outputs:
        # specify an output artifact named "dummy-model2"
        artifacts:
          - name: dummy-model2
            path: /tmp/model
            globalName: dummy-model2
        # we need to define the AICORE_ARTIFACT_SIGNATURES parameter in each
workflow step, where we want to provide
        # signatures for an output artifact
        parameters:
          - name: AICORE_ARTIFACT_SIGNATURES
            valueFrom:
              path: /tmp/signatures.json
      container:
        image: python:3.11-alpine
        imagePullPolicy: Always
```

```

    command: [ "python", "-c" ]
    args:
      - >
        # ... skipped, as it will be the same as in the previous example

```

This workflow template creates two output artifacts: dummy-model1 and dummy-model2, and their name-signature key-value-pairs in JSON format.

## Verifying Signatures from Input Artifacts

If you use input artifacts with signatures in another execution, SAP AI Core provides the signatures of the artifacts as an environment variable named `AICORE_ARTIFACT_SIGNATURES`. The variable is a JSON key-value pair, where each key is the input artifact name and the value is its signature. For example:

```
{
  "name-of-model1": "signature-of-model1",
  "name-of-model2": "signature-of-model2"
}
```

To extract the signature, include the variable `AICORE_ARTIFACT_SIGNATURES` in your workflow template, that returns the value from the key of the pair, for example:

```

apiVersion: argoproj.io/v1alpha1
kind: WorkflowTemplate
metadata:
  name: signature-example-3
  annotations:
    scenarios.ai.sap.com/description: "Example for signature feature"
    scenarios.ai.sap.com/name: "signature-example-scenario"
    executables.ai.sap.com/description: "Input artifact example for signature feature"
    executables.ai.sap.com/name: "signature-example-input-artifacts"
  labels:
    scenarios.ai.sap.com/id: "signature-example-scenario"
    executables.ai.sap.com/id: "signature-example-in-art"
    ai.sap.com/version: "1.0.0"
spec:
  entrypoint: input-artifact-signatures-example
  templates:
    - name: input-artifact-signatures-example
      inputs:
        artifacts:
          - name: dummy-model
            path: /tmp/model
      container:
        image: python:3.11-alpine
        imagePullPolicy: Always
        command: [ "python", "-c" ]
        args:
          - |
            import hashlib, json, os
            # create a hash of the input artifact to compare against the provided signature
            with open("/tmp/model", "rb") as f:
              model_hash_actual = hashlib.file_digest(f,
                "sha256").hexdigest()
            # SAP AI Core will provide the artifact signature via the AICORE_ARTIFACT_SIGNATURES environment variable
            # it will be a string in JSON format
            signatures = json.loads(os.environ["AICORE_ARTIFACT_SIGNATURES"])

```

```
model_hash_expected = signatures["dummy-model"] # the name of the
input artifact is the key
assert model_hash_expected == model_hash_actual, "signatures did
not match"
print("success")
```

This template takes an input artifact with the name `<dummy-model>`, calculates its signature, and compares it to the signature provided by SAP AI Core in the `AICORE_ARTIFACT_SIGNATURES` environment variable. The `AICORE_ARTIFACT_SIGNATURES` environment variable is only set when there is at least one input artifact with a signature, otherwise, it won't be set at all. In a multi-step workflow template, the `AICORE_ARTIFACT_SIGNATURES` environment variable will be supplied to each step of the workflow template, irrespective of where the actual input artifact is defined. In each step, the `AICORE_ARTIFACT_SIGNATURES` variable will hold the signatures of the input artifacts from all steps.

**Parent topic:** [Train Your Model \[page 460\]](#)

## Related Information

[Choose an Instance \[page 462\]](#)  
[Workflow Templates \[page 464\]](#)  
[List Scenarios \[page 469\]](#)  
[List Executables \[page 472\]](#)  
[Create Configurations \[page 478\]](#)  
[List Configurations \[page 480\]](#)  
[Start Training \[page 486\]](#)  
[Stop Training Instances \[page 488\]](#)  
[Delete Training Instances \[page 492\]](#)  
[Efficiency Features \[page 496\]](#)  
[Retrieve Execution Logs \[page 499\]](#)  
[Training Schedules \[page 503\]](#)

## 8.1.2.8 Start Training

**Parent topic:** [Train Your Model \[page 460\]](#)

## Related Information

[Choose an Instance \[page 462\]](#)  
[Workflow Templates \[page 464\]](#)  
[List Scenarios \[page 469\]](#)

[List Executables \[page 472\]](#)  
[Create Configurations \[page 478\]](#)  
[List Configurations \[page 480\]](#)  
[Using Artifact Signatures \[page 482\]](#)  
[Stop Training Instances \[page 488\]](#)  
[Delete Training Instances \[page 492\]](#)  
[Efficiency Features \[page 496\]](#)  
[Retrieve Execution Logs \[page 499\]](#)  
[Training Schedules \[page 503\]](#)

## Using Curl

### Procedure

1. Run the following code:

```
curl --location --request POST '$AI_API_URL/v2/lm/executions' \
```

#### ↔ Output Code

```
{  
  "id": "dea6263e6283321b",  
  "message": "Execution scheduled",  
  "status": "UNKNOWN",  
  "targetStatus": "COMPLETED"  
}
```

2. Check the status of the execution.

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

#### ⚠ Restriction

The number of pods used by executions is limited at tenant level. A tenant is allowed to have at least 50 pods before the quota is enforced. If your tenant reaches this limit, your execution will be queued. You can raise a ticket to increase your quota.

#### ⓘ Note

If the status is dead or pending, there might be errors in the execution. You can check the execution logs for more details, see [Retrieve Execution Logs \[page 499\]](#).

# Using a Third-Party API Platform

## Procedure

- Send a POST request to the endpoint `{apiurl}/v2/lm/executions`. Pass the `configurationId` in the request body.

```
{  
  "configurationId": "47b3eed9-f72f-4a18-b2ab-25b057a3e77f"  
}
```

- Check the status of the execution by submitting a GET request to `{apiurl}/v2/lm/executions/{executionid}`.

Check that the following headers are selected:

Key	Value
Authorization	<code>{{token}}</code>
AI-Resource-Group	<code>{{resource-group}}</code>
Content-Type	<code>application/json</code>

### ⚠ Restriction

The number of pods used by executions is limited at tenant level. A tenant is allowed to have at least 50 pods before the quota is enforced. If your tenant reaches this limit, your execution will be queued. You can raise a ticket to increase your quota.

### ⓘ Note

If the status is dead or pending, there might be errors in the execution. You can check the execution logs for more details, see [Retrieve Execution Logs \[page 499\]](#).

## 8.1.2.9 Stop Training Instances

You can stop a running execution by submitting a PATCH request to `$AI_API_URL/v2/lm/executions/$EXECUTION \.`

### ⓘ Note

Stop is only enabled if the status is running or pending.

[Stop a Single Training Instance \[page 489\]](#)

[Stop Multiple Training Instances \[page 490\]](#)

**Parent topic:** [Train Your Model \[page 460\]](#)

## Related Information

[Choose an Instance \[page 462\]](#)  
[Workflow Templates \[page 464\]](#)  
[List Scenarios \[page 469\]](#)  
[List Executables \[page 472\]](#)  
[Create Configurations \[page 478\]](#)  
[List Configurations \[page 480\]](#)  
[Using Artifact Signatures \[page 482\]](#)  
[Start Training \[page 486\]](#)  
[Delete Training Instances \[page 492\]](#)  
[Efficiency Features \[page 496\]](#)  
[Retrieve Execution Logs \[page 499\]](#)  
[Training Schedules \[page 503\]](#)

### 8.1.2.9.1 Stop a Single Training Instance

You can stop a running execution by submitting a PATCH request to `$AI_API_URL/v2/lm/executions/$EXECUTION \.`

#### ⓘ Note

Stop is only enabled if the status is running or pending.

**Parent topic:** [Stop Training Instances \[page 488\]](#)

## Related Information

[Stop Multiple Training Instances \[page 490\]](#)

## Using Curl

### Procedure

Run the following code:

```
curl --request PATCH $AI_API_URL/v2/lm/executions/$EXECUTION \ --header  
"Authorization: Bearer $TOKEN" --header "AI-Resource-Group: $RESOURCE_GROUP" --  
header "Content-Type: application/json"
```

```
-d '{  
  "targetStatus": "STOPPED"  
}'
```

#### ↔ Output Code

```
{  
  "id": "ee6769e4dc19c0fd",  
  "message": "Execution modification scheduled"  
}
```

## Using a Third-Party API Platform

### Procedure

Send a PATCH request to endpoint {{apiurl}}/v2/lm/executions/{{executionid}} including the execution ID of the execution that you want to stop.

```
{  
  "targetStatus": "STOPPED"  
}
```

### Results

#### ↔ Output Code

```
{  
  "id": "e8c53facc2bfb87a",  
  "message": "Execution modification scheduled"  
}
```

### 8.1.2.9.2 Stop Multiple Training Instances

**Parent topic:** [Stop Training Instances \[page 488\]](#)

### Related Information

[Stop a Single Training Instance \[page 489\]](#)

## Using Curl

### Procedure

Update the request body to:

```
curl --request PATCH - /executions \
--header {"executions": [
{
  "id": "aa97b177-9383-4934-8543-0f91a7a0283a",
  "targetStatus": "STOPPED"
},
{
  "id": "qweq32131-qwee-1231-8543-0f91a7a2e2e",
  "targetStatus": "DELETED"
}
]}
```

### ↔ Output Code

```
{
  "executions": [
    {
      "id": "aa97b177-9383-4934-8543-0f91a7a0283a",
      "message": "Execution modification scheduled"
    },
    {
      "id": "qweq32131-qwee-1231-8543-0f91a7a2e2e",
      "message": "Execution modification scheduled"
    }
  ]
}
```

## Using a Third-Party API Platform

### Procedure

Send a bulk PATCH request to the endpoint: - /executions

Update the request body to:

```
{
  "executions": [
    {
      "id": "aa97b177-9383-4934-8543-0f91a7a0283a",
      "targetStatus": "STOPPED"
    },
    {
      "id": "qweq32131-qwee-1231-8543-0f91a7a2e2e",
      "targetStatus": "DELETED"
    }
  ]
}
```

## ↔ Output Code

```
{  
  "executions": [  
    {  
      "id": "aa97b177-9383-4934-8543-0f91a7a0283a",  
      "message": "Execution modification scheduled"  
    },  
    {  
      "id": "qweq32131-qwee-1231-8543-0f91a7a2e2e",  
      "message": "Execution modification scheduled"  
    }  
  ]  
}
```

### 8.1.2.10 Delete Training Instances

Deleting a training instance releases the SAP AI Core resources that it used.

#### ⚠ Restriction

If your execution is running, you must stop it first. You can stop a running execution by submitting a PATCH request to `$AI_API_URL/v2/lm/executions/$EXECUTION \.`. For more information, see [Stop a Single Training Instance \[page 489\]](#) and [Stop Multiple Training Instances \[page 490\]](#).

[Delete a Single Training Instance \[page 493\]](#)

[Delete Multiple Training Instances \[page 494\]](#)

**Parent topic:** [Train Your Model \[page 460\]](#)

### Related Information

[Choose an Instance \[page 462\]](#)

[Workflow Templates \[page 464\]](#)

[List Scenarios \[page 469\]](#)

[List Executables \[page 472\]](#)

[Create Configurations \[page 478\]](#)

[List Configurations \[page 480\]](#)

[Using Artifact Signatures \[page 482\]](#)

[Start Training \[page 486\]](#)

[Stop Training Instances \[page 488\]](#)

[Efficiency Features \[page 496\]](#)

[Retrieve Execution Logs \[page 499\]](#)

[Training Schedules \[page 503\]](#)

## 8.1.2.10.1 Delete a Single Training Instance

Deleting a training instance releases the SAP AI Core resources that it used.

### ⚠ Restriction

If your execution is running, you must stop it first. You can stop a running execution by submitting a PATCH request to `$AI_API_URL/v2/lm/executions/$EXECUTION \.`. For more information, see [Stop a Single Training Instance \[page 489\]](#) and [Stop Multiple Training Instances \[page 490\]](#).

**Parent topic:** [Delete Training Instances \[page 492\]](#)

## Related Information

[Delete Multiple Training Instances \[page 494\]](#)

## Using Curl

### Procedure

Run the following code:

```
curl --request DELETE $AI_API_URL/v2/lm/executions/$EXECUTION \
    --header "Authorization: Bearer $TOKEN" \
    --header "AI-Resource-Group: $RESOURCE_GROUP"
```

### ↔ Output Code

```
{  
    "id": "ee6769e4dc19c0fd",  
    "message": "Deletion scheduled",  
    "targetStatus": "DELETED"  
}
```

# Using a Third-Party API Platform

## Procedure

Send a DELETE request to {{apiurl}}/v2/lm/executions/{{executionid}}. The header for this request is: AI-Resource-Group: {YOUR-Resource-Group}.

```
{  
    "targetStatus": "DELETED"  
}
```

## Results

### ↔ Output Code

```
{  
    "id": "e8c53facc2bfb87a",  
    "message": "Deletion scheduled",  
    "targetStatus": "DELETED"  
}
```

## 8.1.2.10.2 Delete Multiple Training Instances

Deleting a training instance releases the SAP AI Core resources that it used.

### ⚠ Restriction

If your execution is running, you must stop it first. You can stop a running execution by submitting a PATCH request to \$AI\_API\_URL/v2/lm/executions/\$EXECUTION \. For more information, see [Stop a Single Training Instance \[page 489\]](#) and [Stop Multiple Training Instances \[page 490\]](#).

`bulkUpdates` is a meta capability endpoint of the AI API. It enables or disables bulk PATCH operations. For more information, see [AI API Overview \[page 50\]](#).

The feature is set to false by default. To enable bulk PATCH operations, your template must contain the following snippet, with the relevant values set to `true`.

```
Meta:  
  "bulkUpdates": {  
      "executions": false,  
      "deployments": false  
  }
```

About `bulkUpdates`:

- The maximum number of updates per request is 100.

- Your bulk update can contain a mixture of STOP and DELETE requests.
- Only *running* or *pending* executions or deployments can be stopped.
- Only *stopped*, *dead* or *unknown* executions or deployments can be deleted.
- An ID can only appear once per bulk request. For multiple modifications of the same ID, multiple requests are needed.

**Parent topic:** Delete Training Instances [page 492]

## Related Information

[Delete a Single Training Instance \[page 493\]](#)

# Using Curl

## Procedure

Update the request body to:

```
curl --request PATCH - /executions \
  --header {"executions": [
    {
      "id": "aa97b177-9383-4934-8543-0f91a7a0283a",
      "targetStatus": "STOPPED"
    },
    {
      "id": "qweq32131-qwee-1231-8543-0f91a7a2e2e",
      "targetStatus": "DELETED"
    }
  ]
}
```

### Output Code

```
{
  "executions": [
    {
      "id": "aa97b177-9383-4934-8543-0f91a7a0283a",
      "message": "Execution modification scheduled"
    },
    {
      "id": "qweq32131-qwee-1231-8543-0f91a7a2e2e",
      "message": "Execution modification scheduled"
    }
  ]
}
```

## Using a Third-Party API Platform

### Procedure

Send a bulk PATCH request to the endpoint: – /executions

Update the request body to:

```
{  
  "executions": [  
    {  
      "id": "aa97b177-9383-4934-8543-0f91a7a0283a",  
      "targetStatus": "STOPPED"  
    },  
    {  
      "id": "qweq32131-qwee-1231-8543-0f91a7a2e2e",  
      "targetStatus": "DELETED"  
    }  
  ]  
}
```

#### Output Code

```
{  
  "executions": [  
    {  
      "id": "aa97b177-9383-4934-8543-0f91a7a0283a",  
      "message": "Execution modification scheduled"  
    },  
    {  
      "id": "qweq32131-qwee-1231-8543-0f91a7a2e2e",  
      "message": "Execution modification scheduled"  
    }  
  ]  
}
```

### 8.1.2.11 Efficiency Features

Discover features of the SAP AI Core runtime that improve efficiency and help manage resource consumption.

#### Tenant Warm Node Pool

Tenant warm nodes allow tenants to reserve nodes from specific resource plan types, ensuring that a predefined number of nodes exist in the cluster. The reserved nodes can then be used during model training and serving. Reserving nodes reduces waiting time at workload startup, but incurs costs in line with node use, whether consumed or not.

#### Mechanism of Node Reservation

1. The tenant specifies number of nodes to reserve.

2. An execution or deployment utilizes a reserved node, a replacement reserve node of the same resource plan type is requested from the hyperscaler.

The number of reserved nodes specified by the tenant are consistently available for use.

The minimum number of reserved nodes is 0.

## About Node Reservation

- The number of reserved nodes specified by the tenant are consistently available for use.
- The minimum number of reserved nodes is 0.
- The maximum number of reserved nodes is 10.
- The default number of reserved nodes is 0.

## Reserve Nodes Using a Third-Party API Platform

1. Send a PATCH request to the endpoint `{apiurl}/v2/admin/resources/nodes`
2. Provide the resource plan type and quantity of nodes to reserve in the request body in JSON format:

```
{
  "resourcePlans": [
    {
      "name": "infer.l",
      "request": 1
    },
    {
      "name": "infer.m",
      "request": 1
    },
    {
      "name": "train.l",
      "request": 1
    }
    ...
  ]
}
```

## Reserve Nodes Using curl

Submit a PATCH request to the endpoint `{apiurl}/v2/admin/resources/nodes`

```
curl --request PATCH $AI_API_URL/v2/admin/resources/nodes \
--data-raw '{
  "resourcePlans": [
    {
      "name": "infer.l",
      "request": 1
    },
    {
      "name": "infer.m",
      "request": 1
    },
    {
      "name": "train.l",
      "request": 1
    }
  ]
}'
```

## Check Reserve Node Status Using a Third-Party API Platform

Send a GET request to the endpoint `{apiurl}/v2/admin/resources/nodes`

## Check Reserve Node Status Using curl

```
curl --request GET $AI_API_URL/v2/resources/nodes
```

### ↳ Output Code

```
{
  "resourcePlans": {
    "infer.l": {
      "provisioned": 1,
      "requested": 1
    },
    "infer.m": {
      "provisioned": 1,
      "requested": 1
    },
    "train.l": {
      "provisioned": 1,
      "requested": 1
    }
  }
}
```

- **requested:** the number of reserve nodes requested by the tenant.
- **provisioned:** the number of reserve nodes that are currently present in the cluster.

## Update Quantities of Reserved Nodes

To update the number of nodes reserved, repeat the reservation procedure, with updated quantities in the `request` field.

## Delete Reserved Nodes

To delete reserved nodes, repeat the reservation procedure, with quantities in the `request` field set to `0`.

**Parent topic:** [Train Your Model \[page 460\]](#)

## Related Information

[Choose an Instance \[page 462\]](#)  
[Workflow Templates \[page 464\]](#)  
[List Scenarios \[page 469\]](#)  
[List Executables \[page 472\]](#)  
[Create Configurations \[page 478\]](#)  
[List Configurations \[page 480\]](#)  
[Using Artifact Signatures \[page 482\]](#)  
[Start Training \[page 486\]](#)  
[Stop Training Instances \[page 488\]](#)  
[Delete Training Instances \[page 492\]](#)  
[Retrieve Execution Logs \[page 499\]](#)  
[Training Schedules \[page 503\]](#)

## 8.1.2.12 Retrieve Execution Logs

Deployment and execution logs contain information about API processing and metrics.

### ⓘ Note

All STDOUT from the user containers will be saved by SAP AI Core and be available during log retrieval. As a user of SAP AI Core, If you do not want to keep logs, do not send anything to STDOUT or STDERR. Instead, redirect all output to a file.

You can retrieve the logs for a specific deployment or execution by submitting a GET request. Use the following endpoints to retrieve the logs:

- GET /v2/lm/deployments/{deploymentId}/logs
- GET /v2/lm/executions/{executionId}/logs

Query parameters include:

- `start` : the start time for a query as a string, in RFC 3339-compliant datetime format. Defaults to 1 hour before the current. Example: 2021-05-19T00:00:14.347Z
- `end` : the end time for a query as a string, in RFC 3339-compliant datetime format. Defaults to the current time. Example: 2021-05-19T00:00:14.347Z
- `$top` : the maximum number of entries returned. The default value is 1000; the upper limit is 5000.
- `$order` : the sort order of logs, either `asc` (for ascending, earliest in the order will appear at the top of the list) or `desc` (for descending, most recent in the order will appear at the top of the list). Note the default value is `asc`.

For example:

- `/v2/lm/deployments/{deploymentId}/logs?`  
`start=2021-05-19T00:00:14.347Z&end=2021-05-19T01:00:14.347Z&$top=100&$order=asc`  
- returns the first  
100 lines of a deployment log between **2021-05-19T00:00:14.347Z** and **2021-05-19T01:00:14.347Z**
- `/v2/lm/deployments/{deploymentId}/logs` - returns deployment logs from the preceding hour
- `/v2/lm/executions/{executionId}/logs` - returns execution logs from the preceding hour

## Sample Output

For example, see the following JSON output from the API.

### ↔ Output Code

```
{  
    "data": {  
        "result": [  
            {  
                "container": "storage-initializer",  
                "log": "Container initialized successfully."  
            }  
        ]  
    }  
}
```

```
        "msg": "[I 210531 08:20:51 initializer-entrypoint:13] Initializing, args: src_uri [gs://kserve-samples/models/tensorflow/flowers] dest_path[ [/mnt/models]\n",
      "pod": "tfs-dep-i543026-predictor-default-v6nf5-
deployment-8b58c8ddcfidx",
      "stream": "stderr",
      "timestamp": "2021-05-31T08:20:51.334+00:00"
    },
    {
      "container": "storage-initializer",
      "msg": "[I 210531 08:20:51 storage:45] Copying contents of gs://kserve-samples/models/tensorflow/flowers to local\n",
      "pod": "tfs-dep-i543026-predictor-default-v6nf5-
deployment-8b58c8ddcfidx",
      "stream": "stderr",
      "timestamp": "2021-05-31T08:20:51.335+00:00"
    },
    {
      "container": "storage-initializer",
      "msg": "[W 210531 08:20:51 _metadata:104] Compute Engine Metadata server unavailable onattempt 1 of 3. Reason: [Errno 111] Connection refused\n",
      "pod": "tfs-dep-i543026-predictor-default-v6nf5-
deployment-8b58c8ddcfidx",
      "stream": "stderr",
      "timestamp": "2021-05-31T08:20:51.338+00:00"
    },
    ...
  ]
}
```

**Parent topic:** Train Your Model [page 460]

## Related Information

- [Choose an Instance \[page 462\]](#)
- [Workflow Templates \[page 464\]](#)
- [List Scenarios \[page 469\]](#)
- [List Executables \[page 472\]](#)
- [Create Configurations \[page 478\]](#)
- [List Configurations \[page 480\]](#)
- [Using Artifact Signatures \[page 482\]](#)
- [Start Training \[page 486\]](#)
- [Stop Training Instances \[page 488\]](#)
- [Delete Training Instances \[page 492\]](#)
- [Efficiency Features \[page 496\]](#)
- [Training Schedules \[page 503\]](#)

# Using Curl

## Procedure

Run the following code:

```
curl --request GET "$AI_API_URL/v2/lm/executions/$EXECUTION_ID/logs?  
start=2021-05-19T00:00:14.347Z" --header "Authorization: Bearer $TOKEN" --header  
"AI-Resource-Group: $RESOURCE_GROUP"
```

## Results

### Output Code

```
{
  "data": {
    "result": [
      {
        "container": "storage-initializer",
        "msg": "[I 210531 08:20:51 initializer-entrypoint:13]
Initializing, args: src_uri [gs://kserve-samples/models/tensorflow/flowers]
dest_path[ /mnt/models]\n",
        "pod": "tfs-dep-i543026-predictor-default-v6nf5-
deployment-8b58c8ddcfidx",
        "stream": "stderr",
        "timestamp": "2021-05-31T08:20:51.334+00:00"
      },
      {
        "container": "storage-initializer",
        "msg": "[I 210531 08:20:51 storage:45] Copying contents of
gs://kserve-samples/models/tensorflow/flowers to local\n",
        "pod": "tfs-dep-i543026-predictor-default-v6nf5-
deployment-8b58c8ddcfidx",
        "stream": "stderr",
        "timestamp": "2021-05-31T08:20:51.335+00:00"
      },
      {
        "container": "storage-initializer",
        "msg": "[W 210531 08:20:51 _metadata:104] Compute Engine
Metadata server unavailable onattempt 1 of 3. Reason: [Errno 111] Connection
refused\n",
        "pod": "tfs-dep-i543026-predictor-default-v6nf5-
deployment-8b58c8ddcfidx",
        "stream": "stderr",
        "timestamp": "2021-05-31T08:20:51.338+00:00"
      },
      ...
    ]
  }
}
```

# Using a Third-Party API Platform

## Procedure

- Send a GET request to the endpoint `{apiurl}/v2/1m/executions/{executionId}/logs`.

Check that the following headers are selected:

Key	Value
AI-Resource-Group	<Name of your resource group>

### Output Code

```
{  
    "data": {  
        "result": [  
            {  
                "container": "init",  
                "msg": "time=\"2021-11-22T06:52:27.831Z\" level=info  
msg=\"Starting Workflow Executor\" executorType= version =v3.2.2\\n\",  
                "pod": "ed0f0c1b62945b935"  
                "stream": "stderr"  
                "timestamp": "2021-11-22T06:52:27.831955906+00:00"  
            }  
        ...  
    }  
}
```

- On the *Authorization* tab, set the type to *Bearer Token*.
- Set the token value to `{token}`.
- On the *Header* tab, add the following entry:

Key	Value
AI-Resource-Group	<Name of your resource group>

- Send the request.

## Results

### Output Code

```
{  
    "data": {  
        "result": [  
            {  
                "container": "storage-initializer",  
                "msg": "[I 210531 08:20:51 initializer-entrypoint:13]  
Initializing, args: src_uri [gs://kserve-samples/models/tensorflow/flowers]  
dest_path[ /mnt/models]\\n",  
                "pod": "tfs-dep-i543026-predictor-default-v6nf5-  
deployment-8b58c8ddcf5x",  
                "stream": "stderr",  
            }  
        ...  
    }  
}
```

```

        "timestamp": "2021-05-31T08:20:51.334+00:00"
    },
{
    "container": "storage-initializer",
    "msg": "[I 210531 08:20:51 storage:45] Copying contents of
gs://kserve-samples/models/tensorflow/flowers to local\n",
    "pod": "tfs-dep-i543026-predictor-default-v6nf5-
deployment-8b58c8ddcf5",
    "stream": "stderr",
    "timestamp": "2021-05-31T08:20:51.335+00:00"
},
{
    "container": "storage-initializer",
    "msg": "[W 210531 08:20:51 _metadata:104] Compute Engine
Metadata server unavailable onattempt 1 of 3. Reason: [Errno 111] Connection
refused\n",
    "pod": "tfs-dep-i543026-predictor-default-v6nf5-
deployment-8b58c8ddcf5",
    "stream": "stderr",
    "timestamp": "2021-05-31T08:20:51.338+00:00"
},
...
]
}
}

```

### 8.1.2.13 Training Schedules

In order to run executions periodically, you can define a schedule to start your executions automatically. To do this, you must have prepared your configuration for the execution.

Training schedules have a start and end timestamp. The start timestamp defines when the schedule will first be run to generate executions. The end timestamp defines when the schedule will expire.

A schedule that has not yet expired has ACTIVE status. Upon expiry, the status will change to INACTIVE, and no further executions will start.

The format for these timestamps is defined by RFC3339 section 5.6, without fractions of seconds, for example: 2023-02-09T12:53:47Z. All timestamps are interpreted in UTC time. For more information, see [RFC3339 section 5.6](#) and [AI API Overview \[page 50\]](#).

[Create a Training Schedule \[page 504\]](#)

[List Executions Created by a Training Schedule \[page 505\]](#)

[Change an Existing Training Schedule \[page 507\]](#)

[Delete a Training Schedule \[page 508\]](#)

**Parent topic:** [Train Your Model \[page 460\]](#)

## Related Information

[Choose an Instance \[page 462\]](#)

[Workflow Templates \[page 464\]](#)  
[List Scenarios \[page 469\]](#)  
[List Executables \[page 472\]](#)  
[Create Configurations \[page 478\]](#)  
[List Configurations \[page 480\]](#)  
[Using Artifact Signatures \[page 482\]](#)  
[Start Training \[page 486\]](#)  
[Stop Training Instances \[page 488\]](#)  
[Delete Training Instances \[page 492\]](#)  
[Efficiency Features \[page 496\]](#)  
[Retrieve Execution Logs \[page 499\]](#)

### 8.1.2.13.1 Create a Training Schedule

**Parent topic:** [Training Schedules \[page 503\]](#)

#### Related Information

[List Executions Created by a Training Schedule \[page 505\]](#)  
[Change an Existing Training Schedule \[page 507\]](#)  
[Delete a Training Schedule \[page 508\]](#)

## Using Curl

### Procedure

1. Create a training schedule by submitting a POST request to `$AI_API_URL/v2/lm/executionSchedules`.

In the request body you need to define:

- The schedule in cron format. For more information, see [Cron Formatting](#)
- A name
- The configurationId that you want to use for the training instance
- A start timestamp when the schedule should become active
- An end timestamp when the schedule should become inactive

```
curl --location --request POST "$AI_API_URL/v2/lm/executionSchedules/$EXECUTION_SCHEDULE" \
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP" \
```

```
--data-raw '{ \
  "cron": "0 * * * *",
  "name": "Hourly training run",
  "configurationId": "35b60591-1e48-473b-9b44-d5f8e9e4de32",
  "start": "2023-02-10T10:50:31Z",
  "end": "2023-02-10T12:55:31Z"
}'
```

2. Check the status of the training schedule by submitting a GET request:

```
curl --location --request GET "$AI_API_URL/v2/lm/executionSchedules/
$EXECUTION_SCHEDULE" \\
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP"
```

## Using a Third-Party API Platform

### Procedure

Create a training schedule by submitting a POST request to `{apiurl}/v2/lm/executionSchedules`.

In the request body you need to define:

- The schedule in cron format. For more information, see [Cron Formatting](#)
- A name
- The configurationId that you want to use for the training instance
- A start timestamp when the schedule should become active
- An end timestamp when the schedule should become inactive

```
{
  "cron": "0 * * * *",
  "name": "Hourly training run",
  "configurationId": "36b60691-1e48-473b-9b44-d6f8e9r4de32",
  "start": "2023-02-08T10:50:31Z",
  "end": "2023-02-08T12:56:31Z"
}
```

#### Output Code

```
{
  "id": "799b4e67-a213-40b9-9550-637fde75dbda",
  "message": "Execution Schedule created"
}
```

### 8.1.2.13.2 List Executions Created by a Training Schedule

**Parent topic:** [Training Schedules \[page 503\]](#)

## Related Information

[Create a Training Schedule \[page 504\]](#)

[Change an Existing Training Schedule \[page 507\]](#)

[Delete a Training Schedule \[page 508\]](#)

## Using Curl

### Procedure

Run the following code:

```
curl --location -- request GET "{$AI_API_URL/v2/lm/executionSchedules/$EXECUTION_SCHEDULE" \\ 
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP"
```

## Using a Third-Party API Platform

### Procedure

Get a list of executions created by a training schedule by submitting a GET request to {{apiurl}}/v2/lm/executions?executionScheduleId={{executionScheduleId}}

#### Output Code

```
"count": 2,
"resources": [
{
    "completionTime": "2023-02-08T12:02:56Z",
    "configurationId": "36b60691-1e48-473b-9b44-d6f8e9r4de32",
    "configurationName": "test-for-demo",
    "createdAt": "2023-02-08T12:56:31Z",
    "executionId": "9aea6ade-747d-42eb-ba7c-7dcb617e955a",
    "id": "ef7877dfffb8394be",
    "modifiedAt": "2023-02-08T12:02:48Z",
    "outputArtifacts": [],
    "scenarioId": "platform-test-tensorflow",
    "startTime": "2023-02-08T12:02:48Z",
    "targetStatus": "COMPLETED"
};
...
```

### 8.1.2.13.3 Change an Existing Training Schedule

You can change the cron definition, start, end, and configuration of an existing training schedule.

You cannot change the name field.

**Parent topic:** [Training Schedules \[page 503\]](#)

#### Related Information

[Create a Training Schedule \[page 504\]](#)

[List Executions Created by a Training Schedule \[page 505\]](#)

[Delete a Training Schedule \[page 508\]](#)

## Using Curl

#### Procedure

Submit a PATCH request:

```
curl --location --request PATCH $AI_API_URL/v2/lm/executions?  
executionScheduleId=$EXECUTION_SCHEDULE  
--header "Authorization: Bearer $TOKEN" \  
--header "AI-Resource-Group: $RESOURCE_GROUP" \  
--data-raw '{ "cron": "0 0 * * *" }'
```

## Using a Third-Party API Platform

#### Procedure

Send your changes in a PATCH request to `{apiurl}/v2/lm/executionSchedules/{executionScheduleId}`.

```
{  
  "cron": "0 * * * *",  
}
```

#### Output Code

```
{  
  "id": "799b4e67-a213-40b9-9550-637fde75dbda",
```

```
        "message": "Execution Schedule modified"
    }
```

## Pause or Resume a Training Schedule

The `status` parameter pauses or resumes a schedule:

- `<INACTIVE>`: pauses a schedule.
- `<ACTIVE>`: resumes a schedule.

### 8.1.2.13.4 Delete a Training Schedule

You can delete training schedules that are in states ACTIVE and INACTIVE.

**Parent topic:** [Training Schedules \[page 503\]](#)

#### Related Information

[Create a Training Schedule \[page 504\]](#)

[List Executions Created by a Training Schedule \[page 505\]](#)

[Change an Existing Training Schedule \[page 507\]](#)

## Using Curl

### Context

### Procedure

Submit a DELETE request:

```
curl --location --request DELETE "$AI_API_URL/v2/lm/executionSchedules/
$EXECUTION_SCHEDULE" \\
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP"
```

# Using a Third-Party API Platform

## Context

## Procedure

Send a DELETE request to the endpoint {{apiurl}}/v2/lm/executionSchedules/{{executionScheduleId}}

### ↔ Output Code

```
{  
  "id": "799b4e67-a213-40b9-9550-637fde75dbda",  
  "message": "Execution Schedule deleted"  
}
```

## 8.1.3 Use Your Model

You deploy your AI learning model to run inferences against it.

SAP AI Core provides the means to run AI training and offline batch inference jobs, using the required computational resources (such as GPUs) efficiently.

You can use a model trained in SAP AI Core, or your own pretrained model for deployment and inferencing.

The model serving service includes:

- Serving of AI models through fast, secure, and resource-efficient inference endpoints that can be integrated into online inference scenarios.
- Flexibility to deploy user-supplied Docker images.
- Cost-efficient serving (for example, by autoscaling or serverless support or multimodel serving).
- Parameterized serving templates for main tenants to manage their serving instances.
- Serving instances in resource group namespaces with specific parameters and resource isolation.

The serving templates are used to create model servers. When a model server is up and running, it processes incoming inference requests and returns the results from the AI learning model. Serving templates define how a model is to be deployed. For the model to work, you must provide a spec in line with the [Serving Templates \[page 513\]](#), together with input parameters and artifacts.

Restrict the number of nodes used for processing by specifying parameters minReplicas and maxReplicas.

### Choose an Instance [page 510]

You can configure SAP AI Core to use different infrastructure instances for different tasks, based on demand. SAP AI Core provides several preconfigured infrastructure bundles called “resource plans” and “instance types” for this purpose.

### [Serving Templates \[page 513\]](#)

You use serving templates to manage your serving instances at the level of the main tenant. Serving templates define how a model is to be deployed.

### [List Executables \[page 526\]](#)

An executable is a reusable template that defines a workflow or pipeline for tasks such as training a machine learning model or creating a deployment. It contains placeholders for input artifacts (datasets or models) and parameters (custom key-pair values) that enable the template to be reused in different scenarios.. You can list all of the executables in a resource group and get details of specific executables from a resource group. Serving templates are mapped to deployment executables.

### [Deploy Models \[page 531\]](#)

#### [Inferencing \[page 534\]](#)

#### [Update a Deployment \[page 536\]](#)

#### [Stop Deployments \[page 540\]](#)

#### [Delete Deployments \[page 544\]](#)

#### [Efficiency Features \[page 549\]](#)

Discover features of the SAP AI Core runtime that improve efficiency and help manage resource consumption.

### [Retrieve Deployment Logs \[page 552\]](#)

Deployment and execution logs contain information about API processing and metrics.

**Parent topic:** [ML Operations \[page 450\]](#)

## Related Information

[Connect Your Data \[page 450\]](#)

[Train Your Model \[page 460\]](#)

### 8.1.3.1 Choose an Instance

You can configure SAP AI Core to use different infrastructure instances for different tasks, based on demand. SAP AI Core provides several preconfigured infrastructure bundles called “resource plans” and “instance types” for this purpose.

## Context

You must choose at least one instance. If you select a resource plan, an instance type isn't required and vice versa.

Resource plans and instance types are used to select instances in workflow and serving templates. Different steps of a workflow can have different instances assigned.

In general, if your workload needs GPU acceleration, use one of the GPU-enabled instances. Otherwise, choose a plan based on the anticipated CPU and memory need of your workloads.

Within SAP AI Core, the instances are selected via the `ai.sap.com/resourcePlan` and `ai.sap.com/instanceType` labels at pod level. It maps the selected instance and takes a string value, which is the ID of the instance.

There are limits to the default disk storage size for all these nodes. Datasets that are loaded to the nodes consume disk space. If you have large data sets (larger than 30 GB), or have large models, you may have to increase the disk size. To do so, use the persistent volume claim in Argo Workflows to specify the required disk size (see [Volumes](#) ).

## Instance Types

### ⚠ Restriction

Instance types are only available as part of the “extended” service plan. For more information on instances that can be used with the “standard” service plan.

Within SAP AI Core, the instances are selected via the `ai.sap.com/instanceType` label in your workflow templates. It maps the selected resource and takes a string value, which is the ID of the instance

For information about available instance types and their IDs, see SAP Note [3660109](#), or send a GET request to the following endpoint:

```
{apiurl}/v2/admin/resources/instanceType
```

For example:

### ↔ Sample Code

```
curl GET "$AI_API_URL/v2/admin/resources/instanceType" \
--header "Authorization: Bearer $TOKEN" \
```

## Resource Plans

Within SAP AI Core, the instances are selected via the `ai.sap.com/resourcePlan` label in your workflow templates. It maps the selected resource and takes a string value, which is the ID of the instance.

## Resource Plan Specifications for AWS

Resource Plan IDs	GPUs	CPU Cores	Memory GBs	Code to Allocate instances in Workflow Templates
Starter	-	1	3	ai.sap.com/resourcePlan:starter
Basic	-	3	11	ai.sap.com/resourcePlan:basic
Basic-8x	-	31	116	ai.sap.com/resourcePlan:basic.8x
Train-L	1 V100	7	55	ai.sap.com/resourcePlan:train.l
Infer-S	1 T4	3	10	ai.sap.com/resourcePlan:infer.s
Infer-M	1 T4	7	26	ai.sap.com/resourcePlan:infer.m
Infer-L	1 T4	15	58	ai.sap.com/resourcePlan:infer.l

### ⓘ Note

For the free service plan, only the Starter resource plan is available. Specifying other plans results in an error. For the standard service plan, all resource plans are available.

**Parent topic:** Use Your Model [page 509]

## Related Information

[Serving Templates \[page 513\]](#)

[List Executables \[page 526\]](#)

[Deploy Models \[page 531\]](#)

[Inferencing \[page 534\]](#)

[Update a Deployment \[page 536\]](#)

[Stop Deployments \[page 540\]](#)

[Delete Deployments \[page 544\]](#)

[Efficiency Features \[page 549\]](#)

[Retrieve Deployment Logs \[page 552\]](#)

## Service Usage Reporting

Usage consumption of services is reported in the SAP BTP cockpit on the [Overview](#) page for your global account and on the [Overview](#) and [Usage Analytics](#) pages of your subaccount. The usage report lists usage in billable measures and non-billable measures. Your final monthly bill is based on the billable measure only. Non-billable measures are displayed for reporting purposes only.

### 8.1.3.2 Serving Templates

You use serving templates to manage your serving instances at the level of the main tenant. Serving templates define how a model is to be deployed.

Serving templates are used to deploy one or more trained models on a model server, which process incoming inference requests and returns the results from the machine learning model. Serving templates are stored in your git repository, where you can version them as required.

In SAP AI Core, serving templates are mapped as `executables`. Mapping requires certain attributes in the `metadata` section of your template.

Models are deployed using a simple Kubernetes Custom Resource Definition (CRD), which is provided by KServe. To deploy a model, you must provide a YAML specification that follows the KServe specification and that defines the required input parameters and artifacts.

#### ⚠ Restriction

The maximum number of serving templates is limited at tenant level to 50. If you reach this limit, you receive an error message. To free up space, delete some serving templates. Alternatively, raise a ticket to increase your quota.

To get started, copy the generic serving template below and add your own values as required. You can use any text editor with a YAML plugin to create the templates. Your YAML specification must follow the KServe specs and define the required input parameters and artifacts.

#### Serving Template Parameter Details

Type	Parameter	Description
name (mandatory)		The executable ID. The executable ID must be unique among all executables available within the SAP AI Core main tenant.

Type	Parameter	Description
metadata (mandatory)	name	The technical ID of the serving template that is used to uniquely identify the serving template. It is specified as the serving Template's executable ID, and therefore must be unique among all executables available within the SAP AI Core main tenant.
annotations	scenarios.ai.sap.com/ description(optional)	A description of the scenario to which this executable belongs.
	scenarios.ai.sap.com/name (mandatory)	Scenario name, which is necessary for the AI API to discover a scenario.
	executables.ai.sap.com/ description(optional)	A description of the serving template.
	executables.ai.sap.com/ name(mandatory)	The name of the serving template.
	executables.ai.sap.com/ cascade-update-deployments (optional)	Setting to allow changes to the serving template to be cascaded to associated deployments. See <a href="#">Change Serving Template and Update Deployments [page 525]</a> .
	artifacts.ai.sap.com/ <argo_artifact_name>.desc ription(optional)	You can add further metadata for an artifact using these annotations.
	artifacts.ai.sap.com/ <argo_artifact_name>.label s:   { "ext.ai.sap.com/ customkey1": "customvalue1" , "ext.ai.sap.com/ customkey2": "customvalue2" } (optional)	
labels	scenarios.ai.sap.com/id (mandatory)	The scenario ID to which the serving template belongs
	ai.sap.com/version (mandatory)	The version is intended for customer usage to identify compatible serving template versions.
spec	imagePullSecret(Optional)  Not required if your Docker image is fetched from a public docker repository.	<a href="#">Register Your Docker Registry Secret [page 107]</a> .

Type	Parameter	Description
	minReplicas (Optional)	Deployments run continually by default, but can be stopped manually when inferences are not needed. Setting minReplicas to 0 allows the inference server to enter the stopped state, when not in use. When needed, it is restarted and nodes scaled, based on demand and the min/maxReplicas parameter. For more information see <a href="#">Efficiency Features [page 549]</a> .
	maxReplicas (Optional)	Limit the number of nodes that the inference server scales up to. For more information see <a href="#">Efficiency Features [page 549]</a> .
labels (mandatory)	ai.sap.com/resourcePlan	You must specify the chosen resourcePlan. The value is the string value of the selected resource plan (see <a href="#">Choose an Instance [page 462]</a> ).

## Generic Serving Template

```

apiVersion: ai.sap.com/v1alpha1
kind: ServingTemplate
metadata:
  name: text-clf-infer-tutorial
  annotations:
    scenarios.ai.sap.com/description: "SAP developers tutorial scenario"
    scenarios.ai.sap.com/name: "text-clf-tutorial-scenario"
    executables.ai.sap.com/description: "Inference executable for text classification with Scikit-learn"
    executables.ai.sap.com/name: "text-clf-infer-tutorial-exec"
    artifacts.ai.sap.com/textmodel.kind: "model"
    artifacts.ai.sap.com/textmodel.description: "artifact description"
    artifacts.ai.sap.com/textmodel.labels: |
      {"ext.ai.sap.com/customkey1": "customvalue1", "ext.ai.sap.com/customkey2": "customvalue2"}
    labels:
      ai.sap.com/version: 0.1.0
      scenarios.ai.sap.com/id: "text-clf-tutorial"
spec:
  inputs:
    parameters:
      - name: modelName
        default: value
        type: string
        description: description of the parameter
      - name: image
        default: default-image-value
        type: string
        description: description of the parameter
    artifacts:
      - name: textmodel
  template:
    apiVersion: "serving.kserve.io/v1beta1"

```

```

metadata:
  annotations: |
    autoscaling.knative.dev/metric: rps
    autoscaling.knative.dev/target: 100
    autoscaling.knative.dev/targetBurstCapacity: 70
  labels: |
    ai.sap.com/resourcePlan: starter
spec: |
  predictor:
    imagePullSecrets:
      - name: <Name of your Docker registry secret>
    minReplicas: 0
    maxReplicas: 5
    containers:
      - name: kserve-container
        image: {{inputs.parameters.image}}
    ports:
      - containerPort: 9001
        protocol: TCP
    env:
      - name: STORAGE_URI
        value: "{{inputs.artifacts.textmodel}}"

```

- STORAGE\_URI environment variable name is currently hard-coded in KServe to indicate that the model should be downloaded when a custom predictor is configured with the env var STORAGE\_URI.
- /mnt/models is currently hard-coded in KServe. When you try this example with your own docker container, read the models from that path.

### ⚠ Caution

In the example above, the image parameter is provided as an input. The image value is resolved during rendering, based on priority. {{inputs.parameters.image}} can be rendered from two possible sources, with the following priority:

1. The image value may be specified in configuration/<configuration\_id> when creating the configuration

```

curl --location --request POST '{{apiurl}}/v2/lm/configurations' \
--header 'AI-Resource-Group: default' \
--header 'Authorization: Bearer {TOKEN}' \
--header 'Content-Type: application/json' \
--data-raw '{
  "name": "<configuration_id>",
  "executableId": "<executable_id>",
  "scenarioId": "<scenario_id>",
  "versionId": "0.1.0",
  "parameterBindings": [
    {
      "key": "image",
      "value": "source-1-image-value"
    }
  ],
  "inputArtifactBindings": [
  ]
}'

```

2. The default image value may be specified in the ServingTemplate

```

spec:
  inputs:
    parameters:
      - default: "default-image-value"
        name: image
        type: string

```

If source 1 exists, updating the default value in source 2 will NOT affect the rendered deployment image.

If source 1 does not exist, the deployment image will use (and respond to changes in) the default value from source 2.

## Sync an Application Manually

Applications sync with your GitHub repository automatically at intervals of ~3 minutes. Use the endpoint below to manually request a sync:`{apiurl}/admin/applications/{appName}/refresh`

**Parent topic:** [Use Your Model \[page 509\]](#)

## Related Information

[Choose an Instance \[page 510\]](#)

[List Executables \[page 526\]](#)

[Deploy Models \[page 531\]](#)

[Inferencing \[page 534\]](#)

[Update a Deployment \[page 536\]](#)

[Stop Deployments \[page 540\]](#)

[Delete Deployments \[page 544\]](#)

[Efficiency Features \[page 549\]](#)

[Retrieve Deployment Logs \[page 552\]](#)

## Stop or Delete Multiple Deployments

The feature is set to false by default. To enable bulk PATCH operations, your template must contain the following snippet, with the relevant values set to `true`.

```
Meta:  
  "bulkUpdates": {  
    "executions": false,  
    "deployments": false  
  }
```

## Related Information

[Stop Multiple Deployments \[page 542\]](#)

[Delete Multiple Deployments \[page 547\]](#)

## 8.1.3.2.1 Serving Template API Reference

### API Schema Spec [ai.sap.com/v1alpha1 \[page 518\]](#)

Package `v1alpha1` contains API Schema definitions for the serving `v1alpha1` API group (`ai.sap.com/v1alpha1`).

### KServe Spec [serving.kserve.io/v1beta1 \[page 521\]](#)

Package `v1beta1` contains API Schema definitions for the serving `v1beta1` API group (`serving.kserve.io/v1beta1`).

### 8.1.3.2.1.1 API Schema Spec [ai.sap.com/v1alpha1](#)

Package `v1alpha1` contains API Schema definitions for the serving `v1alpha1` API group (`ai.sap.com/v1alpha1`).

## ServingTemplate

`ServingTemplate` is a type of executable that specifies how a model is to be served.

Field	Description	
<code>metadata</code>	<code>name</code>	(Mandatory) Technical ID of the serving template that is used to uniquely identify the serving template.
<a href="#">Kubernetes meta/v1.ObjectMeta</a>	<code>annotations</code>	Annotations to define the Scenario and Executable info.
	<a href="#">ServingTemplate Metadata Annotations [page 518]</a>	
<code>labels</code>	<code>labels</code>	Labels to define the Scenario and Executable info.
	<a href="#">ServingTemplate Metadata Labels [page 519]</a>	
<code>spec</code>	ServingTemplateSpec contains a template for KServe, input parameters and artifacts required to create a KServe CR.	
<a href="#">ServingTemplate Spec [page 519]</a>		

## ServingTemplate Metadata Annotations

(Appears on [ServingTemplate \[page 518\]](#))

A subset of supported Kubernetes Metadata Annotations

Field	Description
scenarios.ai.sap.com/description	A description of the Scenario (for example, "SAP developers tutorial scenario")
scenarios.ai.sap.com/name	The name of the serving template (for example, "text-clf-tutorial-scenario")
executables.ai.sap.com/description	A description of the serving template. (for example, "Inference executable for text classification with Scikit-learn")
executables.ai.sap.com/name	The name of the serving template (for example, "text-clf-infer-tutorial-exec")

## ServingTemplate Metadata Labels

(Appears on [ServingTemplate \[page 518\]](#))

A subset of supported Kubernetes Metadata Labels

Field	Description
ai.sap.com/version	Version of the Scenario (for example, "1.0.0")
scenarios.ai.sap.com/id	Unique ID of the Scenario (for example, "1234-abcd-efg")

## ServingTemplate Spec

(Appears on [ServingTemplate \[page 518\]](#))

ServingTemplate spec is the top level type for this resource.

Field	Description				
inputs	<table><tr><td>parameters</td><td>Input parameters for the <a href="#">KServe InferenceService Template [page 520]</a></td></tr><tr><td>ServingTemplate Input Parameters [page 520]</td><td></td></tr></table>	parameters	Input parameters for the <a href="#">KServe InferenceService Template [page 520]</a>	ServingTemplate Input Parameters [page 520]	
parameters	Input parameters for the <a href="#">KServe InferenceService Template [page 520]</a>				
ServingTemplate Input Parameters [page 520]					
artifacts	<table><tr><td>Input artifacts for the <a href="#">KServe InferenceService Template [page 520]</a></td></tr><tr><td>ServingTemplate Input Artifacts [page 520]</td></tr></table>	Input artifacts for the <a href="#">KServe InferenceService Template [page 520]</a>	ServingTemplate Input Artifacts [page 520]		
Input artifacts for the <a href="#">KServe InferenceService Template [page 520]</a>					
ServingTemplate Input Artifacts [page 520]					
template	KServe CR template				
	<a href="#">KServe InferenceService Template [page 520]</a>				

## ServingTemplate Input Parameters

(Appears on [ServingTemplate Spec \[page 519\]](#))

Input parameters required for [KServe InferenceService Template \[page 520\]](#) spec

Field	Description
name	Parameter name
default	(Optional) Default value for the parameter
type	(Optional) Type of the parameter. Only “string” type is supported

## ServingTemplate Input Artifacts

(Appears on [ServingTemplate Spec \[page 519\]](#))

Input artifacts required for [KServe InferenceService Template \[page 520\]](#) spec

Field	Description
name	Artifact name

## KServe InferenceService Template

(Appears on: [ServingTemplate Spec \[page 519\]](#))

KServe InferenceService template

Field	Description
apiVersion	KServe API version. For example, <code>serving.kserve.io/v1beta1</code> . String
metadata	Metadata for KServe InferenceService CR. See <a href="#">KServe Metadata Annotations [page 521]</a> and <a href="#">KServe Metadata Labels [page 522]</a> . Annotations and labels in metadata can be parameterized by using <code>{ {inputs.parameters.Parameter-Name-Here} }</code>
spec	A multiline string template for <a href="#">KServe InferenceServiceSpec [page 523]</a> with optional placeholders for parameters and artifacts Multiline string

**Parent topic:** [Serving Template API Reference \[page 518\]](#)

## Related Information

KServe Spec [serving.kserve.io/v1beta1](#) [page 521]

### 8.1.3.2.1.2 KServe Spec [serving.kserve.io/v1beta1](#)

Package v1beta1 contains API Schema definitions for the serving v1beta1 API group ([serving.kserve.io/v1beta1](#)).

#### InferenceService

InferenceService is the schema for the InferenceServices API.

Field	Description		
metadata	Refer to the Kubernetes API documentation for the fields of the metadata field. <a href="#">Kubernetes meta/v1.ObjectMeta</a>		
spec	<a href="#">KServe InferenceServiceSpec</a> [page 523]	predictor <a href="#">KServe PredictorSpec</a> [page 523]	Predictor defines the model serving spec

#### KServe Metadata Annotations

(Appears on [InferenceService](#) [page 521])

A multiline string containing a subset of supported KServe Metadata Labels. You can add a placeholder using {{variable\_name}} to define a variable and replace the value dynamically. For example:

```
autoscaling.knative.dev/target: "{{inputs.parameters.MyAutoScalingTarget}}"
```

Field	Description	Default Value
autoscaling.knative.dev/metric	Metric to be used for autoscaling. Allowed values: <ul style="list-style-type: none"><li>• concurrency</li><li>• rps</li></ul>	concurrency

Field	Description	Default Value
autoscaling.knative.dev/target	Target value for the configured metric. Allowed values: integer	100
autoscaling.knative.dev/targetBurstCapacity	Size of traffic burst deployment can handle without buffering.  Allowed values: Float (0 means requests will only be buffered when number of replicas is 0; -1 means the request is always buffered)	0
autoscaling.knative.dev/window	Amount of time that autoscaler uses before making decisions. The smaller the number, the faster the autoscaler will react.  Allowed values: Integer in seconds (s) or hours (h)	60s
autoscaling.knative.dev/scaleToZeroPodRetentionPeriod	Minimum amount of time that the last pod will remain active after the Autoscaler decides to scale pods to zero.  Allowed values: Non-negative duration string	0s

### Example

```
metadata:
  annotations: |
    autoscaling.knative.dev/metric: rps
    autoscaling.knative.dev/target:
      {{inputs.parameters.MyAutoScalingTarget}}
    autoscaling.knative.dev/targetBurstCapacity: 0
    autoscaling.knative.dev/window: 10s
```

## KServe Metadata Labels

(Appears on [InferenceService \[page 521\]](#))

A multiline string containing a subset of supported KServe Metadata Labels. You can add a placeholder using `{{variable_name}}` to define a variable and replace the resource plan dynamically. For example:

```
ai.sap.com/resourcePlan:
  "{{inputs.parameters.MyResourcePlan}}"
```

Field	Description
ai.sap.com/resourcePlan	Resource Plan to be used in creating the KServe-based Model Server (for example, "basic")

**Example**

```
metadata:
  labels: |
    ai.sap.com/resourcePlan: basic
```

## KServe InferenceServiceSpec

(Appears on [InferenceService \[page 521\]](#))

InferenceServiceSpec is the top level type for this resource.

Field	Description
predictor	Predictor defines the model serving spec <a href="#">KServe PredictorSpec [page 523]</a>

## KServe PredictorSpec

(Appears on [KServe InferenceServiceSpec \[page 523\]](#))

PredictorSpec defines the configuration for a predictor. The following fields follow a "1-of" semantic. Users must specify exactly one spec.

Field	Description
containers	List of containers belonging to the pod. Containers cannot currently be added or removed. There must be at least one container in a pod, cannot be updated. <a href="#">Kubernetes core/v1.Container</a>
imagePullSecrets	List of references to secrets in the same resource group, and used for pulling any of the images used by this spec. The name of the secret must be the same as the one created via Docker Registry Secret creation API. For more information, see <a href="#">Specifying imagePullSecrets on a Pod</a> .

Field	Description
minReplicas (optional) int	Minimum number of replicas. Defaults to 1 but can be set to 0 to enable scale-to-zero.  AI Core will attempt, on best-case effort, to schedule your deployments in different availability zone when you have more than one replica. This way, you can make your deployment High-Available (HA) by configuring min and max-replica accordingly.
maxReplicas (optional) int	Maximum number of replicas for autoscaling. Default: if minReplicas >= 2 then maxReplicas = minReplicas else 2.  AI Core will attempt, on best-case effort, to schedule your deployments in different availability zone when you have more than one replica. This way, you can make your deployment High-Available (HA) by configuring min and max-replica accordingly.
containerConcurrency (optional) int64	Specifies how many requests can be processed concurrently, this sets the hard limit of the container concurrency.  ( <a href="https://knative.dev/docs/serving/autoscaling/concurrency/">https://knative.dev/docs/serving/autoscaling/concurrency/</a> ).
timeout (optional) int64	Specifies the number of seconds to wait before timing out a request to the component.
terminationGracePeriodSeconds (optional) int64	Optional duration in seconds that the pod needs to terminate gracefully, may be decreased in delete request. Value must be non-negative integer. The value zero indicates delete immediately. If this value is nil, the default grace period will be used instead. The grace period is the duration in seconds after the processes running in the pod are sent a termination signal and the time when the processes are forcibly halted with a kill signal. Set this value longer than the expected cleanup time for your process. Defaults to 30 seconds.

## Example

```
spec: |
  predictor:
    imagePullSecrets:
      - name: [DOCKER REGISTRY SECRET GOES HERE]
    minReplicas: 1
    maxReplicas: 5
    containers:
      - name: kserve-container
        image: "[DOCKER IMAGE URL GOES HERE]"
        ports:
```

```
- containerPort: 9001
  protocol: TCP
env:
- name: STORAGE_URI
  value: "{{inputs.artifacts.textmodel}}"
```

**Parent topic:** [Serving Template API Reference \[page 518\]](#)

## Related Information

[API Schema Spec ai.sap.com/v1alpha1 \[page 518\]](#)

### 8.1.3.2.2 Change Serving Template and Update Deployments

If you change a serving template, you can automatically update the deployments that are associated with that template.

#### Prerequisites

The `executables.ai.sap.com/cascade-update-deployments` parameter is present and set to `true` in the serving template. This allows any change to the serving template to trigger an automatic update of the associated deployments.

#### Procedure

Changes to the following are supported:

- `spec.template.spec.default.predictor.minReplicas`
- `spec.template.spec.default.predictor.maxReplicas`
- `spec.template.spec.predictor.containers.image`
- `spec.template.spec.predictor.containers.ports`
- `spec.template.spec.predictor.containers.env`
- Modify a value
- Add a name-value pair
- Parameterized values can be updated to hardcoded and vice versa

### ⚠ Caution

The `image` parameter value can be resolved in multiple ways during rendering, based on priority. For more information, see [Serving Templates \[page 513\]](#).

## Checking the status of updated deployments

You can check the status of the updated deployments by submitting a GET request to `{{apiurl}}/v2/lm/deployments/{{deploymentid}}`

If deployment is updated successfully, `lastOperation` will be updated to "CASCADE-UPDATE".

```
{  
    "createdAt": "2022-02-07T11:20:27+00:00",  
    "deploymentUrl": "https://api.ai.devsg.eu-central-1.mlf-aws-dev.com/v2/inference/deployments/<deployment_id>",  
    "executableId": "<executable_id>",  
    "id": "<deployment_id>",  
    "lastOperation": "CASCADE-UPDATE",  
    "latestRunningTargetId": "<configuration_id>",  
    "modifiedAt": "2022-02-07T11:26:10+00:00",  
    "scenarioId": "<scenario_id>",  
    "status": "RUNNING",  
    "statusDetails": {},  
    "targetId": "<configuration_id>"  
}
```

### 8.1.3.3 List Executables

An executable is a reusable template that defines a workflow or pipeline for tasks such as training a machine learning model or creating a deployment. It contains placeholders for input artifacts (datasets or models) and parameters (custom key-pair values) that enable the template to be reused in different scenarios.. You can list all of the executables in a resource group and get details of specific executables from a resource group. Serving templates are mapped to deployment executables.

**Parent topic:** [Use Your Model \[page 509\]](#)

### Related Information

[Choose an Instance \[page 510\]](#)

[Serving Templates \[page 513\]](#)

[Deploy Models \[page 531\]](#)

[Inferencing \[page 534\]](#)

[Update a Deployment \[page 536\]](#)

[Stop Deployments \[page 540\]](#)

[Delete Deployments \[page 544\]](#)

[Efficiency Features \[page 549\]](#)

[Retrieve Deployment Logs \[page 552\]](#)

## Using Curl

### Procedure

Run the following code:

```
curl --request GET "$AI_API_URL/v2/lm/scenarios" --header "Authorization: Bearer $TOKEN" --header "AI-Resource-Group: $RESOURCE_GROUP"
```

### Results

#### ↔ Output Code

```
{
  "count":4,
  "resources":[
    {
      "createdAt":"2021-02-04T13:11:01+00:00",
      "deployable":true,
      "description":"churn n text class serving executable desc",
      "id":"pytf-serving",
      "inputArtifacts":[
        {
          "name":"model_uri"
        }
      ],
      "labels":[
      ],
      "modifiedAt":"2021-02-04T13:11:01+00:00",
      "name":"churntextclassexecname",
      "parameters":[
        {
          "name":"modelName",
          "type":"string",
          "default": "value",
          "description": "description of the parameter"
        }
      ],
      "scenarioId":"ae0bd260-41ef-4162-81b0-861bd78a8516",
      "versionId":"0.0.1"
    },
    {
      "createdAt":"2021-02-09T07:35:02+00:00",
      "deployable":true,
      "description":"churn n text class serving executable desc",
      "id":"pytf-serving-tracking",
      "inputArtifacts":[
        {
          "name":"textmodel",
        }
      ],
      "labels":[
      ],
      "modifiedAt":"2021-02-09T07:35:02+00:00",
      "name":"churntextclassexecname",
      "parameters":[
        {
          "name":"modelName",
          "type":"string",
          "default": "value",
          "description": "description of the parameter"
        }
      ],
      "scenarioId":"ae0bd260-41ef-4162-81b0-861bd78a8516",
      "versionId":"0.0.1"
    }
  ]
}
```

```

        "kind": "model",
        "description": "artifact description",
        "labels": [
            {
                "key": "ext.ai.sap.com/customkey1",
                "value": "customvalue1"
            },
            {
                "key": "ext.ai.sap.com/customkey2",
                "value": "customvalue2"
            }
        ]
    },
    "labels": [
        ],
        "modifiedAt": "2021-02-09T07:35:02+00:00",
        "name": "churntextclassexecname",
        "parameters": [
            {
                "name": "modelName",
                "type": "string"
            }
        ],
        "scenarioId": "ae0bd260-41ef-4162-81b0-861bd78a8516",
        "versionId": "0.0.1"
    },
    {
        "createdAt": "2021-02-09T07:35:03+00:00",
        "deployable": false,
        "description": "churn and text class executable desc",
        "id": "pytf-training-tracking",
        "inputArtifacts": [
            {
                "name": "churn-data"
            },
            {
                "name": "textclass-data"
            }
        ],
        "labels": [
            ],
            "modifiedAt": "2021-02-09T07:35:03+00:00",
            "name": "churnntextclassexecutablename",
            "outputArtifacts": [
                {
                    "name": "churn-pickle",
                    "kind": "model",
                    "description": "artifact description",
                    "labels": [
                        {
                            "key": "ext.ai.sap.com/customkey1",
                            "value": "customvalue1"
                        },
                        {
                            "key": "ext.ai.sap.com/customkey2",
                            "value": "customvalue2"
                        }
                    ]
                },
                {
                    "name": "pytf-model"
                }
            ],
            "parameters": [
                {

```

```

        "name": "train-epoch",
        "type": "string"
    }
],
"scenarioId": "ae0bd260-41ef-4162-81b0-861bd78a8516",
"versionId": "0.0.1"
},
{
    "createdAt": "2021-02-04T14:11:02+00:00",
    "deployable": false,
    "description": "churn and text class executable desc",
    "id": "test-training",
    "inputArtifacts": [
        {
            "name": "churn-data"
        },
        {
            "name": "textclass-data"
        }
    ],
    "labels": [
        ],
    "modifiedAt": "2021-02-04T14:11:02+00:00",
    "name": "churnntextclassexecutablename",
    "outputArtifacts": [
        {
            "name": "churn-pickle"
        },
        {
            "name": "pytf-model"
        }
    ],
    "parameters": [
        {
            "name": "train-epoch",
            "type": "string"
        }
    ],
    "scenarioId": "ae0bd260-41ef-4162-81b0-861bd78a8516",
    "versionId": "0.0.1"
}
]
}

```

### ⓘ Note

The `<modifiedAt>` field denotes the timestamp of the latest successful sync. The output `1970-01-01T00:00:00+00:00` indicates an error.

## Using a Third-Party API Platform

### Procedure

1. Add your scenario ID as the value for the `scenarioId` environment variable.
2. Send a GET request to the endpoint `{apiurl}/v2/lm/scenarios/{scenarioId}/executables`
3. On the *Authorization* tab, set the type to *Bearer Token*.

- Set the token value to `{token}`.
- On the [Header](#) tab, add the following entry:

Key	Value
AI-Resource-Group	<Name of your resourceGroup> default is used)

- Send the request.

#### ↔ Output Code

```
{
  "count": 3,
  "resources": [
    {
      "createdAt": "2021-10-07T20:07:18+00:00",
      "deployable": true,
      "description": "Inference executable for text classification with Scikit-learn",
      "id": "text-clf-infer-tutorial",
      "inputArtifacts": [
        ...
      ]
    }
  ]
}
```

#### ⓘ Note

The `<modifiedAt>` field denotes the timestamp of the latest successful sync. The output `1970-01-01T00:00:00+00:00` indicates an error.

## Get Executable Details with curl

### Procedure

Run the following code:

```
curl --request GET "$AI_API_URL/v2/lm/scenarios" --header "Authorization: Bearer $TOKEN" --header "AI-Resource-Group: $RESOURCE_GROUP"
```

#### ↔ Output Code

```
{
  "createdAt": "2021-02-04T14:11:02+00:00",
  "deployable": false,
  "description": "churn and text class executable desc",
  "id": "test-training",
  "inputArtifacts": [
    {
      "name": "churn-data"
    },
    {
      "name": "textclass-data"
    }
  ],
  "labels": [
```

```

] ,
"modifiedAt": "2021-02-04T14:11:02+00:00",
"name": "churnntextclassexecutablename",
"outputArtifacts": [
    {
        "name": "churn-pickle"
    },
    {
        "name": "pytf-model"
    }
],
"parameters": [
    {
        "name": "train-epoch",
        "type": "string"
    }
],
"scenarioId": "ae0bd260-41ef-4162-81b0-861bd78a8516",
"versionId": "0.0.1"
}

```

### ⓘ Note

The `<modifiedAt>` field denotes the timestamp of the latest successful sync. The output `1970-01-01T00:00:00+00:00` indicates an error.

## Get Executable Details with a Third-Party API Platform

### Procedure

1. Add the environment variable `executableid` and as its value, enter the ID of the executable.
2. Send a GET request to the endpoint `{apiurl}/v2/lm/scenarios/{scenarioId}/executables/{executableId}`
3. On the *Authorization* tab, set the type to *Bearer Token*.
4. Set the token value to `{token}`.
5. On the *Header* tab, add the following entry:

Key	Value
AI-Resource-Group	<code>&lt;Name of your resourceGroup&gt;</code> (in the example, default is used)

### 8.1.3.4 Deploy Models

## Related Information

[Choose an Instance \[page 510\]](#)  
[Serving Templates \[page 513\]](#)  
[List Executables \[page 526\]](#)  
[Inferencing \[page 534\]](#)  
[Update a Deployment \[page 536\]](#)  
[Stop Deployments \[page 540\]](#)  
[Delete Deployments \[page 544\]](#)  
[Efficiency Features \[page 549\]](#)  
[Retrieve Deployment Logs \[page 552\]](#)

## Using Curl

### Procedure

1. Trigger the deployment.

```
curl --request POST $AI_API_URL/v2/lm/deployments \
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP"
--data-raw '{
  "configurationId": " 2b72d740-5a89-4cf7-b37c-85973eed6ae "
}'
```

#### Output Code

```
{
  "deploymentUrl": "",
  "id": "dda5d19065d5b1f4",
  "message": "Deployment created.",
  "status": "UNKNOWN"
}
```

2. Note the deploy environment variable for later use.
3. Check the status of the deployment.

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

#### ⓘ Note

Configurations are checked when POST requests are submitted. If a configuration is incorrect configuration, an error message outlining the details will be returned..

If the status of the deployment is dead or pending after successful submission, there might be errors in the deployment. You can check the deployment logs for more details, see [Retrieve Deployment Logs \[page 552\]](#).

### ↔ Output Code

```
{  
  "configurationExecutableId": "hello-tf-1-15",  
  "configurationId": "2b72d740-5a89-4cf7-b37c-85973eed6ae",  
  "configurationName": "hello-tf-1-15-config",  
  "createdAt": "2020-11-11T06:34:24Z",  
  "createdBy": "user",  
  "deploymentUrl": "https://my-deployment-url.com",  
  "id": "d291766fd1072b3f",  
  "modifiedAt": "2020-11-11T06:37:29Z",  
  "modifiedBy": "user",  
  "scenarioId": "dba85cf3-2d69-498d-8ce5-4a415c9116dc",  
  "status": "RUNNING",  
  "targetStatus": "RUNNING",  
  "versionId": "0.1.0"  
}
```

## Using a Third-Party API Platform

### Procedure

1. Send a POST request to the endpoint {{apiurl}}/v2/lm/deployments.
2. Pass the configurationId in the request body.

```
{  
  "deploymentUrl": "",  
  "id": "d94771b082c5cbcc",  
  "message": "Deployment scheduled.",  
  "status": "UNKNOWN"  
}
```

3. Check the status of the deployment by sending a GET request to {{apiurl}}/v2/lm/deployments/{{deploymentid}}.

```
{  
  "deploymentUrl": "your deployment url",  
  "modifiedAt": "2021-09-27T10:48:51Z",  
  "scenarioId": "b5379278-887c-4156-9a65-a6c11d6f1a71",  
  "startTime": "2021-09-27T10:33:12Z",  
  "status": "RUNNING"  
}
```

### ⓘ Note

Configurations are checked when POST requests are submitted. If a configuration is incorrect configuration, an error message outlining the details will be returned..

If the status of the deployment is dead or pending after successful submission, there might be errors in the deployment. You can check the deployment logs for more details, see [Retrieve Deployment Logs \[page 552\]](#).

# Parameters and Quotas

## Optional Parameters

The duration of a deployment can be limited using the `ttl` parameter. It takes an integer for quantity, and a single letter to specify units of time. Only minutes (`m`), hours (`h`) and days (`d`), are supported, and values must be natural numbers. For example, "`ttl`": "5h" gives the deployment a duration of 5 hours. `4.5h` and `4h30m` are not valid inputs. If no value is passed, the duration of the deployment is indefinite. Once the duration expires, the deployment is stopped and deleted.

## Deployment Quotas

Each tenant is assigned a default quota that limits the number of deployments and replicas per deployment. If you reach this quota, your deployment will not be created, and you will be notified. You can free up your quota by deleting existing deployments.

Alternatively, you can request a quota increase by creating a ticket. The component name is **CA-ML-AIC** and ticket title is **Request to Increase Quota**.

### 8.1.3.5 Inferencing

Use the URL from your model deployment to access the results of your model.

To inference your model, send a POST request to "`$DEPLOYMENT_URL/<path that you have defined>`". For example, TensorFlow models follow the path `v1/models/$MODEL_NAME:predict`.

**Parent topic:** [Use Your Model \[page 509\]](#)

## Related Information

[Choose an Instance \[page 510\]](#)

[Serving Templates \[page 513\]](#)

[List Executables \[page 526\]](#)

[Deploy Models \[page 531\]](#)

[Update a Deployment \[page 536\]](#)

[Stop Deployments \[page 540\]](#)

[Delete Deployments \[page 544\]](#)

[Efficiency Features \[page 549\]](#)

[Retrieve Deployment Logs \[page 552\]](#)

# Using Curl

## Prerequisites

- The deployment must have the state “pending”, “running” or “dead”.
- The new configuration contains the same `scenarioId` and `executableId` as the currently active configuration.

### ⓘ Note

Dead deployments can only be patched within 7 days. After 7 days from the time that the deployment reached DEAD status, it will be deleted and will no longer be available.

## Context

Use the URL from your model deployment to access the results of your model.

To inference your model, send a POST request to `{apiurl}/v2/lm/deployments`. For example, TensorFlow models follow the path `v1/models/MODEL_NAME:predict`.

## Procedure

In this example, the name of the model is “churn”.

```
curl --location --request POST '$deploymentUrl/v1/models/churn:predict' \
```

# Using a Third-Party API Platform

## Prerequisites

- The deployment must have the state “pending”, “running” or “dead”.
- The new configuration contains the same `scenarioId` and `executableId` as the currently active configuration.

### ⓘ Note

Dead deployments can only be patched within 7 days. After 7 days from the time that the deployment reached DEAD status, it will be deleted and will no longer be available.

## Context

Use the URL from your model deployment to access the results of your model.

To inference your model, send a POST request to `{apiurl}/v2/lm/deployments`. For example, TensorFlow models follow the path `v1/models/MODEL_NAME:predict`.

## Procedure

As the deployment URL, pass the URL that was returned in the response body in the `{apiurl}/v2/lm/deployments` call (see [Deploy Models \[page 531\]](#)).

As the request body, enter a sample instance in JSON format.

In this example, the name of the model is “churn”.

```
POST {deploymenturl}/v1/models/churn:predict

{
  "region": 3,
  "tenure": 37,
  "age": 53,
  "address": 13,
  "income": 48,
  "employ": 10,
  "gender": 0,
  "equip": 0,
  ...
}
```

### 8.1.3.6 Update a Deployment

You can update a deployment with a new configuration while retaining the inference URL.

During the transition to the new deployment configuration, the inference requests will continue to work.

When the new configuration is deployed, the deployment may end in a “dead” status, or get stuck in the “pending” status, and never get to a “running” state, due to incorrect configuration. In this case, inference requests will no longer work. The last known running configuration ID is documented in field `latestRunningConfigurationId` and can be used in another PATCH request to return to the last running configuration.

If the updated deployment reaches the “running” state, `latestRunningConfigurationId` will be updated to the new configuration.

**Parent topic:** [Use Your Model \[page 509\]](#)

## Related Information

[Choose an Instance \[page 510\]](#)  
[Serving Templates \[page 513\]](#)  
[List Executables \[page 526\]](#)  
[Deploy Models \[page 531\]](#)  
[Inferencing \[page 534\]](#)  
[Stop Deployments \[page 540\]](#)  
[Delete Deployments \[page 544\]](#)  
[Efficiency Features \[page 549\]](#)  
[Retrieve Deployment Logs \[page 552\]](#)

## Using Curl

### Context

You can update a deployment with a new configuration while retaining the inference URL.

During the transition to the new deployment configuration, the inference requests will continue to work.

When the new configuration is deployed, the deployment may end in a “dead” status, or get stuck in the “pending” status, and never get to a “running” state, due to incorrect configuration. In this case, inference requests will no longer work. The last known running configuration ID is documented in field *latestRunningConfigurationId* and can be used in another PATCH request to return to the last running configuration.

If the updated deployment reaches the “running” state, *latestRunningConfigurationId* will be updated to the new configuration.

#### ⓘ Note

Dead deployments can only be patched within 7 days. After 7 days from the time that the deployment reached DEAD status, it will be deleted and will no longer be available.

### Procedure

1. Update the deployment by submitting a PATCH request to `{apiurl}/v2/lm/deployments/{{deploymentid}}`.
2. Pass the new `configurationId` in the request body.

```
curl --request PATCH $AI_API_URL/v2/lm/deployments/$DEPLOYMENT_ID \
--header "Authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP" \
--header 'Content-Type: application/json' \
--data-raw '{
```

```
        "configurationId": "490a02b0-4b97-48e8-b905-1c6ae5ea5b1c"
    }'
```

#### ↔ Output Code

```
{
  "id": "d748fdae9f88a9b0",
  "message": "Deployment modification scheduled"
}'
```

3. Check the status of the deployment by submitting a GET request to `{apiurl}/v2/lm/deployments/{deploymentid}`.

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

#### ↔ Output Code

```
{
  "configurationId": "490a02b0-4b97-48e8-b905-1c6ae5ea5b1c",
  "configurationName": "mnist-serving",
  "createdAt": "2021-12-01T10:20:28Z",
  "deploymentUrl": "https://my-deployment-url.com",
  "id": "d748fdae9f88a9b0",
  "lastOperation": "UPDATE",
  "latestRunningConfigurationId": "490a02b0-4b97-48e8-b905-1c6ae5ea5b1c",
  "modifiedAt": "2021-12-01T11:19:21Z",
  "scenarioId": "platform-test-mnist-tensorflow",
  "startTime": "2021-12-01T10:47:23Z",
  "status": "PENDING",
  "statusDetails": {},
  "submissionTime": "2021-12-01T11:03:55Z",
  "targetStatus": "RUNNING"
}'
```

## Using a Third-Party API Platform

### Context

You can update a deployment with a new configuration while retaining the inference URL.

During the transition to the new deployment configuration, the inference requests will continue to work.

When the new configuration is deployed, the deployment may end in a “dead” status, or get stuck in the “pending” status, and never get to a “running” state, due to incorrect configuration. In this case, inference requests will no longer work. The last known running configuration ID is documented in field `latestRunningConfigurationId` and can be used in another PATCH request to return to the last running configuration.

If the updated deployment reaches the “running” state, `latestRunningConfigurationId` will be updated to the new configuration.

### ⓘ Note

Dead deployments can only be patched within 7 days. After 7 days from the time that the deployment reached DEAD status, it will be deleted and will no longer be available.

## Procedure

1. Update the deployment by submitting a PATCH request to `{apiurl}/v2/lm/deployments/{deploymentid}`.
2. Pass the new `configurationId` in the request body.

```
{configurationId: "58c4f650-2494-4786-aa28-e2ab413d545"}
```

### ⇢ Output Code

```
{
  "id": "d748fd9e9f88a9b0",
  "message": "Deployment modification scheduled"
}
```

3. Check the status of the deployment by submitting a GET request to `{apiurl}/v2/lm/deployments/{deploymentid}`.

### ⇢ Output Code

```
{
  "configurationId": "58v4f650-4794-4786-aa28-e2ab4136d45",
  "configurationName": "mnist-serving",
  "createdAt": "2021-12-01T10:20:28Z",
  "deploymentUrl": "https://<url>/inference/deployments/d748fd9e9f88a9b0",
  "id": "d748fd9e9f88a9b0",
  "lastOperation": "UPDATE",
  "latestRunningConfigurationId": "58b4f650-2494-4786-aa28-e2ab4136d545",
  "modifiedAt": "2021-12-01T10:48:36Z",
  "scenarioId": "platform-test-mnist-tensorflow",
  "startTime": "2021-12-01T10:47:23Z",
  "status": "RUNNING",
  "statusDetailing": {
    "deployment_info": [
      {
        "last_transition_time": "2021-12-01T10:46:15Z",
        "message": null,
        "status": "True",
        "type": "IngressReady"
      },
      ...
    ]
  }
}
```

## 8.1.3.7 Stop Deployments

Stopping a deployment releases the SAP AI Core runtime computing resources that it used. A stopped deployment does not incur costs.

### ⓘ Note

Stop is only enabled if the status is running or pending.

[Stop a Single Deployment \[page 540\]](#)

[Stop Multiple Deployments \[page 542\]](#)

**Parent topic:** [Use Your Model \[page 509\]](#)

## Related Information

[Choose an Instance \[page 510\]](#)

[Serving Templates \[page 513\]](#)

[List Executables \[page 526\]](#)

[Deploy Models \[page 531\]](#)

[Inferencing \[page 534\]](#)

[Update a Deployment \[page 536\]](#)

[Delete Deployments \[page 544\]](#)

[Efficiency Features \[page 549\]](#)

[Retrieve Deployment Logs \[page 552\]](#)

## 8.1.3.7.1 Stop a Single Deployment

Stopping a deployment releases the SAP AI Core runtime computing resources that it used. A stopped deployment does not incur costs.

### ⓘ Note

Stop is only enabled if the status is running or pending.

**Parent topic:** [Stop Deployments \[page 540\]](#)

## Related Information

[Stop Multiple Deployments \[page 542\]](#)

## Using Curl

### Procedure

Update the deployment by submitting a PATCH request to {{apiurl}}/v2/lm/deployments/{{deploymentid}}.

Update the request body to:

```
curl --request PATCH $AI_API_URL/v2/lm/deployments/$DEPLOYMENT_ID \
--header "authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP" \
--header 'content-Type: application/json' \
--data-raw '{
  "targetStatus": "STOPPED"
}'
```

#### Output Code

```
{
  "id": "d748fdae9f88a9b0",
  "message": "Deployment modification scheduled"
}
```

## Next Steps

Check the status of the deployment by submitting a GET request to {{apiurl}}/v2/lm/deployments/{{deploymentid}}.

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

# Using a Third-Party API Platform

## Procedure

Stop the deployment by submitting a PATCH request to `{apiurl}/v2/lm/deployments/{deploymentid}`. The header for this request is: `AI-Resource-Group: {YOUR-Resource-Group}`. The Body for this request is:

```
{  
  "targetStatus": "STOPPED"  
}
```

The screenshot shows the POSTMAN interface with a PATCH request to `/lm/deployments/d509aa095937801e`. The Body tab contains the JSON payload: 

```
1 {  
2   "targetStatus": "STOPPED"  
3 }
```

. The response tab shows a 202 Accepted status with the message: 

```
1 {  
2   "id": "d509aa095937801e",  
3   "message": "Deployment modification scheduled"  
4 }
```

.

## Next Steps

Check the status of the deployment by submitting a GET request to `{apiurl}/v2/lm/deployments/{deploymentid}`.

## 8.1.3.7.2 Stop Multiple Deployments

Stopping a deployment releases the SAP AI Core runtime computing resources that it used. A stopped deployment does not incur costs.

### ⓘ Note

Stop is only enabled if the status is running or pending.

`bulkUpdates` is a meta capability endpoint of the AI API. It enables or disables bulk PATCH operations. For more information, see [AI API Overview \[page 50\]](#).

The feature is set to false by default. To enable bulk PATCH operations, your template must contain the following snippet, with the relevant values set to `true`.

```
Meta:  
  "bulkUpdates": {  
    "executions": false,  
    "deployments": false  
  }
```

About `bulkUpdates`:

- The maximum number of updates per request is 100.
- Your bulk update can contain a mixture of STOP and DELETE requests.
- Only `running` or `pending` executions or deployments can be stopped.
- Only `stopped`, `dead` or `unknown` executions or deployments can be deleted.
- An ID can only appear once per bulk request. For multiple modifications of the same ID, multiple requests are needed.

**Parent topic:** [Stop Deployments \[page 540\]](#)

## Related Information

[Stop a Single Deployment \[page 540\]](#)

# Using Curl

## Procedure

Send a bulk PATCH request to the endpoint: `- /deployments`

Update the request body to:

```
{  
  "executions": [  
    {  
      "id": "aa97b177-9383-4934-8543-0f91a7a0283a",  
      "targetStatus": "STOPPED"  
    },  
    {  
      "id": "qweq32131-qwee-1231-8543-0f91a7a2e2e",  
      "targetStatus": "DELETED"  
    }  
  ]  
}
```

### ↔ Output Code

```
{  
  "executions": [  
    {
```

```
        "id": "aa97b177-9383-4934-8543-0f91a7a0283a",
        "message": "Execution modification scheduled"
    },
    {
        "id": "qweq32131-qwee-1231-8543-0f91a7a2e2e",
        "message": "Execution modification scheduled"
    }
]
```

## Using a Third-Party API Platform

### Procedure

Send a bulk PATCH request to the endpoint: - /deployments

Update the request body to:

```
{
  "executions": [
    {
      "id": "aa97b177-9383-4934-8543-0f91a7a0283a",
      "targetStatus": "STOPPED"
    },
    {
      "id": "qweq32131-qwee-1231-8543-0f91a7a2e2e",
      "targetStatus": "DELETED"
    }
  ]
}
```

### Output Code

```
{
  "executions": [
    {
        "id": "aa97b177-9383-4934-8543-0f91a7a0283a",
        "message": "Execution modification scheduled"
    },
    {
        "id": "qweq32131-qwee-1231-8543-0f91a7a2e2e",
        "message": "Execution modification scheduled"
    }
  ]
}
```

### 8.1.3.8 Delete Deployments

Deleting a deployment releases the SAP AI Core resources that it used.

### **⚠ Restriction**

If your deployment is running, you must stop it first. You can stop a deployment by submitting a PATCH request to `{apiurl}/v2/lm/deployments/{deploymentid}`. For more information, see [Stop Deployments \[page 540\]](#)

[Delete a Single Deployment \[page 545\]](#)

[Delete Multiple Deployments \[page 547\]](#)

**Parent topic:** [Use Your Model \[page 509\]](#)

## **Related Information**

[Choose an Instance \[page 510\]](#)

[Serving Templates \[page 513\]](#)

[List Executables \[page 526\]](#)

[Deploy Models \[page 531\]](#)

[Inferencing \[page 534\]](#)

[Update a Deployment \[page 536\]](#)

[Stop Deployments \[page 540\]](#)

[Efficiency Features \[page 549\]](#)

[Retrieve Deployment Logs \[page 552\]](#)

### **8.1.3.8.1 Delete a Single Deployment**

Deleting a deployment releases the SAP AI Core resources that it used.

### **⚠ Restriction**

If your deployment is running, you must stop it first. You can stop a deployment by submitting a PATCH request to `{apiurl}/v2/lm/deployments/{deploymentid}`. For more information, see [Stop a Single Deployment \[page 540\]](#)

**Parent topic:** [Delete Deployments \[page 544\]](#)

## **Related Information**

[Delete Multiple Deployments \[page 547\]](#)

## Using Curl

### Procedure

1. Update the deployment by submitting a DELETE request to {{apiurl}}/v2/lm/deployments/{{deploymentid}}.
2. Update the request body to:

```
curl --request DELETE $AI_API_URL/v2/lm/deployments/$DEPLOYMENT_ID \
--header "authorization: Bearer $TOKEN" \
--header "AI-Resource-Group: $RESOURCE_GROUP" \
--header 'content-Type: application/json' \
--data-raw '{
  "targetStatus": "DELETED"
}'
```

#### Output Code

```
{
  "id": "d748fdae9f88a9b0",
  "message": "Deployment modification scheduled"
}
```

3. Check the status of the deployment by submitting a GET request to {{apiurl}}/v2/lm/deployments/{{deploymentid}}.

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

## Using a Third-Party API Platform

### Procedure

Delete the deployment by submitting a PATCH request to {{apiurl}}/v2/lm/deployments/{{deploymentid}}. The header for this request is: AI-Resource-Group: {YOUR-Resource-Group}. The Body for this request is:

```
{
  "targetStatus": "DELETED"
}
```

### Next Steps

Check the status of the deployment by submitting a GET request to {{apiurl}}/v2/lm/deployments/{{deploymentid}}.

## 8.1.3.8.2 Delete Multiple Deployments

Deleting a deployment releases the SAP AI Core resources that it used.

### ⚠ Restriction

If your deployment is running, you must stop it first. You can stop a deployment by submitting a PATCH request to {{apiurl1}}/v2/lm/deployments/{{deploymentid}}. For more information, see [Stop Multiple Deployments \[page 542\]](#).

`bulkUpdates` is a meta capability endpoint of the AI API. It enables or disables bulk PATCH operations. For more information, see [AI API Overview \[page 50\]](#).

The feature is set to `false` by default. To enable bulk PATCH operations, your template must contain the following snippet, with the relevant values set to `true`.

```
Meta:  
  "bulkUpdates": {  
    "executions": false,  
    "deployments": false  
  }
```

About `bulkUpdates`:

- The maximum number of updates per request is 100.
- Your bulk update can contain a mixture of STOP and DELETE requests.
- Only *running* or *pending* executions or deployments can be stopped.
- Only *stopped*, *dead* or *unknown* executions or deployments can be deleted.
- An ID can only appear once per bulk request. For multiple modifications of the same ID, multiple requests are needed.

**Parent topic:** [Delete Deployments \[page 544\]](#)

## Related Information

[Delete a Single Deployment \[page 545\]](#)

[AI API Overview \[page 50\]](#)

## Using Curl

### Procedure

Send a bulk PATCH request to the endpoint: – /deployments

Update the request body to:

```
{  
  "executions": [  
    {  
      "id": "aa97b177-9383-4934-8543-0f91a7a0283a",  
      "targetStatus": "STOPPED"  
    },  
    {  
      "id": "qweq32131-qwee-1231-8543-0f91a7a2e2e",  
      "targetStatus": "DELETED"  
    }  
  ]  
}
```

#### ↔ Output Code

```
{  
  "executions": [  
    {  
      "id": "aa97b177-9383-4934-8543-0f91a7a0283a",  
      "message": "Execution modification scheduled"  
    },  
    {  
      "id": "qweq32131-qwee-1231-8543-0f91a7a2e2e",  
      "message": "Execution modification scheduled"  
    }  
  ]  
}
```

## Using a Third-Party API Platform

### Procedure

Send a bulk PATCH request to the endpoint: - /deployments

Update the request body to:

```
{  
  "executions": [  
    {  
      "id": "aa97b177-9383-4934-8543-0f91a7a0283a",  
      "targetStatus": "STOPPED"  
    },  
    {  
      "id": "qweq32131-qwee-1231-8543-0f91a7a2e2e",  
      "targetStatus": "DELETED"  
    }  
  ]  
}
```

#### ↔ Output Code

```
{  
  "executions": [  
    {  
      "id": "aa97b177-9383-4934-8543-0f91a7a0283a",  
      "targetStatus": "STOPPED"  
    }  
  ]  
}
```

```
        "message": "Execution modification scheduled"
    },
{
    "id": "qweq32131-qwee-1231-8543-0f91a7a2e2e",
    "message": "Execution modification scheduled"
}
]
```

### 8.1.3.9 Efficiency Features

Discover features of the SAP AI Core runtime that improve efficiency and help manage resource consumption.

#### Autoscaling

SAP AI Core includes parameters to reduce the number of nodes used based on current consumption, or impose usage limits during periods of high consumption. These parameters allow your workload the flexibility to scale based on demand, and for consumption to be capped, limiting your consumption and therefore costs. For more information, see [Serving Templates](#).

#### Scaling to 0

Where non-uniform loads are expected, scaling to 0 allows nodes to enter a sleeping state when demand allows, limiting your consumption, and therefore costs. Nodes wake when demand increases, which has an increased response time. The Global Node pool reduces this cold start time. For more information, see [Serving Templates](#).

#### Global Node Pool

When the inference server scales up from a sleeping state, there is some additional response time. To reduce this, SAP AI Core has a Global Node Pool, which keeps commonly used nodes reserved to allow for shorter response times. You do not need to do anything to make use of the Global Node Pool, it is already in place. To reduce response times further, avoid a cold start altogether by setting your autoscaling parameter to 1. For more information, see [Serving Templates](#).

## Scaling to 1

Cold starts can be avoided completely by scaling to 1. This keeps a single node warm, even when it is not needed, reducing response time. However, it does not offer the consumption and cost savings associated with scaling to 0. For more information, see [Serving Templates](#).

## Duration

The default duration is indefinite, however the `ttl` parameter limits the duration of a deployment to minutes, hours, or days. This parameter allows you to plan the deletion if your model servers and model deployment URL, allowing for an expected period of use and avoiding unnecessary consumption and costs afterwards. For more information, see [Deploy Models](#) and [About the AI API](#).

## Tenant Warm Node Pool

Tenant warm nodes allow tenants to reserve nodes from specific resource plan types, ensuring that a predefined number of nodes exist in the cluster. The reserved nodes can then be used during model training and serving. Reserving nodes reduces waiting time at workload startup, but incurs costs in line with node use, whether consumed or not.

### Mechanism of Node Reservation

1. The tenant specifies number of nodes to reserve.
2. An execution or deployment utilizes a reserved node, a replacement reserve node of the same resource plan type is requested from the hyperscaler.

The number of reserved nodes specified by the tenant are consistently available for use.

The minimum number of reserved nodes is 0.

### About Node Reservation

- The number of reserved nodes specified by the tenant are consistently available for use.
- The minimum number of reserved nodes is 0.
- The maximum number of reserved nodes is 10.
- The default number of reserved nodes is 0.

## Reserve Nodes Using a Third-Party API Platform

1. Send a PATCH request to the endpoint `{apiurl}/v2/admin/resources/nodes`
2. Provide the resource plan type and quantity of nodes to reserve in the request body in JSON format:

```
{  
  "resourcePlans": [  
    {  
      "name": "infer.1",  
      "request": 1  
    },  
    {  
      "name": "infer.1",  
      "request": 1  
    }  
  ]  
}
```

```

        "name": "infer.m",
        "request": 1
    },
    {
        "name": "train.l",
        "request": 1
    }
    ...
]
}

```

## Reserve Nodes Using curl

Submit a PATCH request to the endpoint {{apiurl}}/v2/admin/resources/nodes

```

curl --request PATCH $AI_API_URL/v2/admin/resources/nodes \
--data-raw '{
    "resourcePlans": [
        {
            "name": "infer.l",
            "request": 1
        },
        {
            "name": "infer.m",
            "request": 1
        },
        {
            "name": "train.l",
            "request": 1
        }
    ]
}'

```

## Check Reserve Node Status Using a Third-Party API Platform

Send a GET request to the endpoint {{apiurl}}/v2/admin/resources/nodes

## Check Reserve Node Status Using curl

```
curl --request GET $AI_API_URL/v2/resources/nodes
```

### Output Code

```
{
    "resourcePlans": {
        "infer.l": {
            "provisioned": 1,
            "requested": 1
        },
        "infer.m": {
            "provisioned": 1,
            "requested": 1
        },
        "train.l": {
            "provisioned": 1,
            "requested": 1
        }
    }
}
```

- **requested:** the number of reserve nodes requested by the tenant.

- `provisioned`: the number of reserve nodes that are currently present in the cluster.

## Update Quantities of Reserved Nodes

To update the number of nodes reserved, repeat the reservation procedure, with updated quantities in the `request` field.

## Delete Reserved Nodes

To delete reserved nodes, repeat the reservation procedure, with quantities in the `request` field set to `0`.

**Parent topic:** [Use Your Model \[page 509\]](#)

## Related Information

[Choose an Instance \[page 510\]](#)

[Serving Templates \[page 513\]](#)

[List Executables \[page 526\]](#)

[Deploy Models \[page 531\]](#)

[Inferencing \[page 534\]](#)

[Update a Deployment \[page 536\]](#)

[Stop Deployments \[page 540\]](#)

[Delete Deployments \[page 544\]](#)

[Retrieve Deployment Logs \[page 552\]](#)

[Service Plans \[page 64\]](#)

### 8.1.3.10 Retrieve Deployment Logs

Deployment and execution logs contain information about API processing and metrics.

You can retrieve the logs for a specific deployment or execution by submitting a GET request. Use the following endpoints to retrieve the logs:

- `GET /v2/lm/deployments/{deploymentId}/logs`
- `GET /v2/lm/executions/{executionId}/logs`

Query parameters include:

- `start` : the start time for a query as a string, in RFC 3339-compliant datetime format. Defaults to 1 hour before the current. Example: `2021-05-19T00:00:14.347Z`
- `end` : the end time for a query as a string, in RFC 3339-compliant datetime format. Defaults to the current time. Example: `2021-05-19T00:00:14.347Z`
- `$top` : the maximum number of entries returned. The default value is 1000; the upper limit is 5000.
- `$order` : the sort order of logs, either `asc` (for ascending, earliest in the order will appear at the top of the list) or `desc` (for descending, most recent in the order will appear at the top of the list). Note the default value is `asc`.

For example:

- `/v2/lm/deployments/{deploymentId}/logs?`  
`start=2021-05-19T00:00:14.347Z&end=2021-05-19T01:00:14.347Z&$top=100&$order=asc`  
- returns the first  
100 lines of a deployment log between **2021-05-19T00:00:14.347Z** and **2021-05-19T01:00:14.347Z**
- `/v2/lm/deployments/{deploymentId}/logs` - returns deployment logs from the preceding hour
- `/v2/lm/executions/{executionId}/logs` - returns execution logs from the preceding hour

**Parent topic:** [Use Your Model \[page 509\]](#)

## Related Information

[Choose an Instance \[page 510\]](#)

[Serving Templates \[page 513\]](#)

[List Executables \[page 526\]](#)

[Deploy Models \[page 531\]](#)

[Inferencing \[page 534\]](#)

[Update a Deployment \[page 536\]](#)

[Stop Deployments \[page 540\]](#)

[Delete Deployments \[page 544\]](#)

[Efficiency Features \[page 549\]](#)

## Using Curl

### Procedure

Run the following code:

```
curl --request GET "$AI_API_URL/v2/lm/deployments/$DEPLOYMENT_ID/logs?  
start=2021-05-19T00:00:14.347Z" --header "Authorization: Bearer $TOKEN" --header  
"AI-Resource-Group: $RESOURCE_GROUP"
```

## Using a Third-Party API Platform

### Procedure

1. Create a new GET request and enter the URL `{apiurl}/v2/lm/deployments/{deploymentid}/logs`.

Key	Value
AI-Resource-Group	<Name of your resource group>
	<pre>{   "data": {     "result": [       {         "container": "storage-initializer",         "msg": "[I 211121 12:36:59 initializer- entrypoint:13] Initializing, args: src_uri [s3://hcpe597ff51-40f5-42c9- a75a-744281742e61*manji1234/4dfead0ec09e716a/text-model-tutorial] dest_path [ [/mnt/models/n]],         "pod": "d757b72bcd373305-predictor-default-nblx6- deployment*sc4cdc46cfmw",         "stream": "stderr",         "timestamp": "2021-11-22T06:52:27.831955906+00:00"       }     ...   } }</pre>

2. On the *Authorization* tab, set the type to *Bearer Token*.
3. Set the token value to `{token}`.
4. On the *Header* tab, add the following entry:

Key	Value
AI-Resource-Group	<Name of your resourceGroup> (in the example, default is used)

5. Send the request.

## Sample Output

For example, see the following JSON output from the API.

### ↔ Output Code

```
{
  "data": {
    "result": [
      {
        "container": "storage-initializer",
        "msg": "[I 210531 08:20:51 initializer-entrypoint:13]
Initializing, args: src_uri [gs://kserve-samples/models/tensorflow/flowers]
dest_path[ [/mnt/models]\n",
        "pod": "tfs-dep-i543026-predictor-default-v6nf5-
deployment-8b58c8ddcfidx",
        "stream": "stderr",
        "timestamp": "2021-05-31T08:20:51.334+00:00"
      },
      {
        "container": "storage-initializer",
        "msg": "[I 210531 08:20:51 predictor-v6nf5-
deployment-8b58c8ddcfidx initialized]"
      }
    ]
  }
}
```

```
        "msg": "[I 210531 08:20:51 storage:45] Copying contents of  
gs://kserve-samples/models/tensorflow/flowers to local\n",  
        "pod": "tfs-dep-i543026-predictor-default-v6nf5-  
deployment-8b58c8ddcfidx",  
        "stream": "stderr",  
        "timestamp": "2021-05-31T08:20:51.335+00:00"  
    },  
    {  
        "container": "storage-initializer",  
        "msg": "[W 210531 08:20:51 _metadata:104] Compute Engine  
Metadata server unavailable onattempt 1 of 3. Reason: [Errno 111] Connection  
refused\n",  
        "pod": "tfs-dep-i543026-predictor-default-v6nf5-  
deployment-8b58c8ddcfidx",  
        "stream": "stderr",  
        "timestamp": "2021-05-31T08:20:51.338+00:00"  
    },  
    ...  
}  
]
```

## 8.2 Metrics

The AI API provides the ability to track metrics, and to customize or filter which metrics are reported.

**Parent topic:** Predictive AI [\[page 450\]](#)

### Related Information

[ML Operations](#) [\[page 450\]](#)

#### 8.2.1 Metrics Tracking with AI API

You can use AI API to track and fetch metrics for executions and models. In addition, metrics can be compared using the SAP AI Launchpad interface.

### Usage Authorization

The following role collections are required to use the AI API to track metrics.

Function	Role Collection
Create Metrics	aicore_admin or aicore_scenario_editor
View Metrics	aicore_viewer or aicore_scenario_viewer

### 8.2.1.1 Get Metrics

## Getting Metrics with a Third-Party API Platform

### Procedure

1. Prepare a GET request to the endpoint {{apiurl}}/v2/lm/metrics with the following query parameters and headers:

#### Query Parameters

Parameter	Required	Data Type	Description
executionIds	Yes	Array of String	ID of execution for which to list metrics information.
\$select	No	Array of String	<p>Projects only the resources that are specified in the \$select parameter, and executionId.</p> <p>The supported values of \$select are <b>metrics</b>, <b>tags</b> and <b>customInfo</b> or any combination of these and <b>*</b>.</p> <p>If the value is <b>*</b>, all metric resource data is returned.</p>

#### Header

Field	Required	Data Type	Description
AI-Resource-Group	Yes	String	ID of the resource group that contains the execution.

2. On the *Authorization* tab, set the type to *Bearer Token*.
3. Set the token value to `{ {token} }`.
4. Set the query parameters in *Params*.

Key	Value
\$Select	metrics
executionIds	<Your execution ID>

5. Set the resource group *Header*.

Key	Value
AI-Resource-Group	<Your Resource Group name>

6. Send the request.

## Results

You will receive the following response:

JSON Field	Description
customInfo	Array of dictionary. Each dictionary is a contains custom information in string.
executionId	ID of the execution, which the metrics belong to.
metrics	Array of dictionary of metric.
tags	

### ↔ Output Code

```
{
  "count": 2,
  "resources": [
    {
      "customInfo": [-
        ],
      "executionID": "e3de6dbbca8621b5",
      "metrics": [-
        ]
    },
    {
      "customInfo": [-
        ],
      "executionID": "edbbb3c7a4ba64b9",
      "metrics": [-
        ]
    }
  ]
}
```

### *ⓘ* Note

It is possible to use more than one metric in the \$select parameter by separating valid entries with a comma.

Key	Value
\$Select	metrics
executionIds	<An execution ID>,<Another execution ID>

## Getting Metrics with curl

The following demonstrates how you can manually track metrics information, and patch and delete metrics.

### Procedure

```
curl --location -g --request GET '$AI_API_URL/v2/lm/metrics?  
$select=metrics,tags,customInfo&executionIds=e3de6dbbca8621c5,ed4bb3c7a4ba64b9'  
--header 'AI-Resource-Group: default' \  
--header 'Authorization: Bearer $TOKEN'
```

### *ⓘ* Note

It is possible to use any one of the allowed values or combination of the allowed values (metrics, tags, customInfo) or \* in the \$select parameter.

## Results

### *↔* Output Code

```
{  
    "count": 2,  
    "resources": [  
        {  
            "customInfo": [  
                {  
                    "name": "confusion matrix",  
                    "value": "[{'Predicted': 'False', 'Actual': 'False','value':  
34},{{'Predicted': 'False', 'Actual': 'True', 'value': 124}, {'Predicted':  
'True', 'Actual': 'False', 'value': 165},{ 'Predicted': 'True','Actual':  
'True', 'value': 36}]"  
                }  
            ],  
            "executionId": "ec83b8e837fe4a56",  
            "metrics": [  
                {
```

```

    "labels": [
        {
            "name": "m11",
            "value": "alpha"
        },
        {
            "name": "metrics.ai.sap.com/Artifact.name",
            "value": "text-model-tutorial"
        }
    ],
    "name": "Error Rate",
    "step": 2,
    "timestamp": "2020-09-29T11:40:10.330000Z",
    "value": 0.98
},
],
"tags": [
    {
        "name": "text-model-tutorial",
        "value": "beta-3"
    }
]
},
{
"customInfo": [
    {
        "name": "confusion matrix",
        "value": " precision recall f1-score support\n\n 0 0.85 0.97 0.91\n273\\n 1 0.96 0.79 0.87 227\\n\\n accuracy 0.89 500\\n macro avg 0.90 0.88 0.89\n500\\nweighted avg 0.90 0.89 0.89 500\\n"
    }
],
"executionId": "ea4af58c56c26384",
"metrics": [
    {
        "labels": [
            {
                "name": "train",
                "value": "range 0-400"
            }
        ],
        "name": "Accuracy",
        "step": 1,
        "timestamp": "2021-12-03T06:23:57.945336Z",
        "value": 0.9675
    },
    {
        "labels": [
            {
                "name": "train",
                "value": "range 400-800"
            }
        ],
        "name": "Accuracy",
        "step": 2,
        "timestamp": "2021-12-03T06:23:58.089152Z",
        "value": 0.94
    },
    {
        "labels": [
            {
                "name": "train",
                "value": "range 800-1200"
            }
        ],
        "name": "Accuracy",
        "step": 3,
        "timestamp": "2021-12-03T06:23:58.195216Z",
        "value": 0.93
    }
]
}

```

```

        "value": 0.9625
    },
    {
        "labels": [
            {
                "name": "train",
                "value": "range 1200-1600"
            }
        ],
        "name": "Accuracy",
        "step": 4,
        "timestamp": "2021-12-03T06:23:58.306945Z",
        "value": 0.9425
    },
    {
        "labels": [
            {
                "name": "train",
                "value": "range 1600-2000"
            }
        ],
        "name": "Accuracy",
        "step": 5,
        "timestamp": "2021-12-03T06:23:58.421745Z",
        "value": 0.945
    },
    {
        "labels": [
            {
                "name": "metrics.ai.sap.com/Artifact.name",
                "value": "text-model-tutorial"
            }
        ],
        "name": "Error Rate",
        "step": 0,
        "timestamp": "2021-12-03T06:23:58.569476Z",
        "value": 0.1099999999999999
    },
    {
        "labels": [],
        "name": "n_compliments",
        "step": 0,
        "timestamp": "2021-12-03T06:23:57.576040Z",
        "value": 1173.0
    },
    {
        "labels": [],
        "name": "n_complaints",
        "step": 0,
        "timestamp": "2021-12-03T06:23:57.575618Z",
        "value": 1327.0
    },
    {
        "labels": [],
        "name": "n_samples",
        "step": 0,
        "timestamp": "2021-12-03T06:23:57.574852Z",
        "value": 2500.0
    },
    {
        "labels": [
            {
                "name": "train",
                "value": "80%"
            },
            {
                "name": "step",
                "value": "preprocessing"
            }
        ]
    }
]
```

```

        }
    ],
    "name": "split_samples",
    "step": 0,
    "timestamp": "2021-12-03T06:23:57.757594Z",
    "value": 2000.0
},
{
    "labels": [
        {
            "name": "validation",
            "value": "20%"
        },
        {
            "name": "step",
            "value": "preprocessing"
        }
    ],
    "name": "split_samples",
    "step": 0,
    "timestamp": "2021-12-03T06:23:57.757608Z",
    "value": 500.0
}
],
"tags": [
    {
        "name": "Reading Model",
        "value": "OK"
    },
    {
        "name": "Test Inference",
        "value": "checked"
    }
]
}
]
}

```

### 8.2.1.2 Patch Metrics

## Patching Metrics Using a Third-Party API Platform

### Procedure

1. Prepare a PATCH request to the endpoint {{apiurl1}}/v2/lm/metrics with the following headers:

Header

Field	Required	Data Type	Description
AI-Resource-Group	Yes	String	ID of the resource group that contains the execution.

Field	Required	Data Type	Description
Content-Type	Yes	value = application/merge-patch+json	

2. On the *Authorization* tab, set the type to *Bearer Token*.
3. Set the token value to `{${token}}`.
4. Set the *Header*.

Key	Value
AI-Resource-Group	<Your resource group name>
Content-Type	<Your content type>

5. Add the *Body*.

#### ↔ Sample Code

```
{
  "executionId": "${{executionid}}",
  "metrics": [
    {
      "name": "Error Rate",
      "value": 0.98,
      "timestamp": "2021-06-28T07:50:24.589Z",
      "step": 2,
      "labels": [
        {
          "name": "group",
          "value": "tree-82"
        }
      ]
    ],
    "tags": [
      {
        "name": "Artifact Group",
        "value": "RFC-1"
      }
    ],
    "customInfo": [
      {
        "name": "Confusion Matrix",
        "value": "[{'Predicted': 'False', 'Actual': 'False', 'value': 34}, {'Predicted': 'False', 'Actual': 'True', 'value': 124}, {'Predicted': 'True', 'Actual': 'False', 'value': 165}, {'Predicted': 'True', 'Actual': 'True', 'value': 36}]"
      }
    ]
  }
}
```

6. Send the request.

## Results

As the response, you will receive a 204 (no content) response.

## Patching Metrics Using curl

### Procedure

Run the following code:

```
curl --location --request GET '$AI_API_URL/v2/lm/metrics?
executionId=e1c49497ccf6dde8' \
--header 'AI-Resource-Group: default' \
--header 'Authorization: Bearer $TOKEN'
\
--data-raw '{
"executionId": "e1c49497ccf6dde8",
"metrics": [
{
"name": "Error Rate",
"value": 0.98,
"timestamp": "2021-06-28T07:50:24.589Z",
"step": 2,
"labels": [
{
"name": "group",
"value": "tree-82"
}
]
},
{
"tags": [
{
"name": "Artifact Group",
"value": "RFC-1"
}
],
"customInfo": [
{
"name": "Confusion Matrix",
"value": "[{\\"Predicted\\": \\"False\\", \\"Actual\\": \\"False\\", \\"value\\": 34}, {\\"Predicted\\": \\"False\\", \\"Actual\\": \\"True\\", \\"value\\": 124}, {\\"Predicted\\": \\"True\\", \\"Actual\\": \\"False\\", \\"value\\": 165}, {\\"Predicted\\": \\"True\\", \\"Actual\\": \\"True\\", \\"value\\": 36}]"
}
]
}
}'
```

### 8.2.1.3 Query Metric Data

You can query metrics by submitting a GET request to the endpoint `/v2/lm/metrics`. To fetch the tracking data, you can use the following parameters:

- `AI-Resource-Group` (header) – string – UUID
- `executionIds` (query) – string – Retrieve metrics based on up to 10 execution IDs (comma-separated list)
- `$select` (query) – Selectively retrieve metric resource data, such as metrics and custom info. For example, if the parameter value is a wildcard (`*`), then all metric resource data is returned. If the parameter

value is `custom info`, then only custom info data is returned. Values should be entered as an array. Only the values `metrics`, `tags`, `customInfo` and `*` are supported.

## Examples

### Responses of GET API

In this example, the `$select` parameter value is a wildcard (\*), so all metric resource data is returned.

Response Code	Description
200	List of tracking metadata, where each item includes metrics, labels, parameters, and tags.

#### Sample Code

```
{
  "count": 1,
  "resources": [
    {
      "executionId": "aa97b177-9383-4934-8543-0f91a7a0283a",
      "metrics": [
        {
          "name": "Error Rate",
          "value": 0.98,
          "timestamp": "2020-11-12T12:36:13.730Z",
          "step": 2,
          "labels": [
            {
              "name": "group",
              "value": "tree-82"
            }
          ]
        }
      ],
      "tags": [
        {
          "name": "Artifact Group",
          "value": "RFC-1"
        }
      ],
      "customInfo": [
        {
          "name": "Confusion Matrix",
          "value": "[{'Predicted': 'False', 'Actual': 'False', 'value': 34}, {'Predicted': 'False', 'Actual': 'True', 'value': 124}, {'Predicted': 'True', 'Actual': 'False', 'value': 165}, {'Predicted': 'True', 'Actual': 'True', 'value': 36}]"
        }
      ]
    }
  ]
}
```

Response Code	Description
400	The specification of the resource was incorrect.

#### ↔ Sample Code

```
{  
  "error": {  
    "code": "<error code>",  
    "message": "<error message>.",  
    "requestId": "9832bf934f3743v3948v3",  
    "target": "/metrics",  
    "details": [  
      {  
        "code": "<error code>",  
        "message": "<error message>."  
      }  
    ]  
  }  
}
```

#### Response Code

#### Description

501

The operation is not supported.

#### ↔ Sample Code

```
{  
  "error": {  
    "code": "02010055",  
    "message": "Metrics was not found.",  
    "requestId": "9832bf934f3743v3948v3",  
    "target": "/metrics",  
    "details": [  
      {  
        "code": "9827389374",  
        "message": "Empty result set."  
      }  
    ]  
  }  
}
```

## 8.2.1.4 Store Metric Data

You can store metrics by submitting a PATCH request to the endpoint `/v2/1m/metrics`. To store tracking data, you must provide the parameter `AI-Resource-Group` header string.

The `executionId`, `metrics` labels, tags, and `customInfo` together form a `MetricResource` that is used as the request body for the PATCH API to persist tracking data.

The following limits apply to each of the attributes in the `MetricResource`:

- `metrics` – maximum 1000
- `labels` – maximum 20, with a maximum character limit of 256 for each label
- `tags` – maximum 100, with a maximum character limit of 256 for each tag
- `customInfo` – maximum 100, with a maximum size of 5 MB for the entire `customInfo` object

## → Recommendation

Do not use the tracking functions provided by SAP AI Core to track sensitive information. For more information, see [Security \[page 587\]](#).

## executionId

A unique identifier for an execution.

## metrics

A metric is a key/value pair where the value is numeric. Metrics are used to measure and monitor the performance, progress, and quality of a model during the training process. Common examples of metrics include accuracy, precision, recall, F1 score, and mean squared error. Metrics can have optional step, timestamp, and label fields, which provide additional information about the metric's context. Every metric (and associated labels), tag, and custom info must be associated with a specific training execution. This association allows for proper organization, tracking, and comparison of different training runs. Once the metric, tag, and custom info are saved, they can be queried using an execution ID.

Metrics Attributes

Attribute Name	Description	Type	Constraints
Name (mandatory)	Name of the metric	String	Pattern: [\w-]{1,64} Maxlength: 256
Value (mandatory)	Numeric value of the metric	Number	
Timestamp	Time when the metric was created or logged in RFC3339 format	String (\$date-time)	
Step	Step is an optional integer that represents any measurement of training progress (number of training iterations, number of epochs, and so on) for the metric	Integer	Minimum: 0
Labels	A list of name-value object pairs associated with some metric.	See <a href="#">Label Attributes [page 567]</a>	

## labels

A label is a key-value pair attached to a metric to provide additional context and metadata. Labels are used to classify and categorize metrics, enabling users to filter, group, and aggregate data for better insights and analysis. A set of labels can be applied to each instance of a metric record, allowing for more granular and targeted monitoring.

### Label Attributes

Attribute Name	Description	Type	Constraints
Name (mandatory)	Name of the label	String	Minlength: 1 Maxlength: 256
Value (mandatory)	Value of the label	String	Minlength: 1 Maxlength: 256

For a metric to be associated with a model, you must explicitly associate it with a model artifact:

### Sample Code

```
"labels": [
  {
    "name": "group",
    "value": "tree-82"
  },
  {
    "name": "metrics.ai.sap.com/Artifact.name",
    "value": "my_model_name"
  }
]
```

When an execution produces only one output model artifact, all metrics captured in the execution are associated with that output artifact. If an execution produces more than one output model artifact, each metric captured must be explicitly associated with a model artifact (as shown in the example above).

## tags

A tag is a name/value pair that is used to categorize and organize test executions. Tags allow for the segregation and grouping of test executions based on specific criteria, making it easier to manage, analyze, and report on the results. For example, you can assign tags to a group of selected test executions to indicate their purpose, priority, or other relevant characteristics. A set of tags can be associated with a `MetricResource`, which is an entity that represents a specific metric or measurement. The `MetricResource`, in turn, is linked to an execution, establishing a connection between the tags and the corresponding test run. This relationship enables the effective tracking, monitoring, and evaluation of test executions based on their assigned tags.

#### Tag Attributes

Attribute Name	Description	Type	Constraints
Name (mandatory)	Name of the tag	String	Minlength: 1 Maxlength: 256
Value (mandatory)	Value of the tag	String	Minlength: 1 Maxlength: 256

#### customInfo

Custom info key/value pairs that enable the capture of large amounts of metadata, typically associated with an execution. Custom info provides rendering or semantic information regarding a metric for a consuming application or complex metrics in JSON format.

A set of such custom info objects can be associated with a MetricResource.

#### CustomInfo Attributes

Attribute Name	Description	Type	Constraints
Name (mandatory)	Name of the CustomInfo	String	Minlength: 1 Maxlength: 256
Value (mandatory)	Value of the CustomInfo	String	Minlength: 1 Maxlength: 256

## Examples

### Example Request Body for PATCH API

```
{  
  "executionId": "aa97b177-9383-4934-8543-0f91a7a0283a",  
  "metrics": [  
    {  
      "name": "Error Rate",  
      "value": 0.98,  
      "timestamp": "2020-11-12T12:18:01.539Z",  
      "step": 2,  
      "labels": [  
        {  
          "name": "group",  
          "value": "tree-82"  
        },  
        {  
          "name": "metrics.ai.sap.com/Artifact.name",  
          "value": "my_model_name"  
        }  
      ]  
    }  
  ]  
}
```

```

    "tags": [
      {
        "name": "Artifact Group",
        "value": "RFC-1"
      }
    ],
    "customInfo": [
      {
        "name": "Confusion Matrix",
        "value": "[{'Predicted': 'False', 'Actual': 'False', 'value': 34}, {'Predicted': 'False', 'Actual': 'True', 'value': 124}, {'Predicted': 'True', 'Actual': 'False', 'value': 165}, {'Predicted': 'True', 'Actual': 'True', 'value': 36}]"
      }
    ]
  }
}

```

## Responses of PATCH API

Response Code	Description
204	Metrics was successfully updated/created
400	The specification of the resource was incorrect

### ↔ Sample Code

```
{
  "error": {
    "code": "<error code>",
    "message": "<error message>.",
    "requestId": "9832bf934f3743v3948v3",
    "target": "/metrics",
    "details": [
      {
        "code": "<error code>",
        "message": "<error message>"
      }
    ]
  }
}
```

Response Code	Description
413	Request entity is larger than limits defined by server.

### ↔ Sample Code

```
{
  "error": {
    "code": "02000005",
    "message": "PayloadLimitException",
    "requestId": "9832bf934f3743v3948v3",
    "target": "/metrics",
    "details": [
      {
        "code": "02000005",
        "message": "PayloadLimitException"
      }
    ]
  }
}
```

## 8.2.1.5 Delete Metrics

You can delete metrics by submitting a DELETE request to the endpoint `/v2/lm/metrics`. To delete tracking data, you must provide the following parameters:

- `AI-Resource-Group` (header) – string
- `executionId` (query)

Query Parameters

Parameter	Required	Data Type	Description
<code>executionIds</code>	Yes	Array of String	ID of execution

Header

Field	Required	Data Type	Description
<code>AI-Resource-Group</code>	Yes	String	ID of the resource group that contains the execution.

## Responses of DELETE API

Response Code	Description
200	Metric resource was successfully deleted
404	The specified resource was not found

### ↔ Sample Code

Sample 404 response:

```
{  
  "error": {  
    "code": "02010055",  
    "message": "Metrics was not found.",  
    "requestId": "9832bf934f3743v3948v3",  
    "target": "/metrics",  
    "details": [  
      {  
        "code": "9827389374",  
        "message": "Empty result set."  
      }  
    ]  
  }  
}
```

# Deleting Metrics with a Third-Party API Platform

## Procedure

1. Send a DELETE request to the endpoint {{apiurl}}/v2/lm/metrics
2. On the *Authorization* tab, set the type to *Bearer Token*.
3. Set the token value to {{token}}.
4. Set the query parameters in *Params*.

Key	Value
executionId	{{executionId}}

5. Set the *Header*.

Key	Value
AI-Resource-Group	<Your Resource Group name>

6. Send the request.

## Results

You should receive the message Metric Resource was successfully deleted.

# Deleting Metrics with curl

## Procedure

Send a DELETE request to the endpoint {{apiurl}}/v2/lm/metrics with the required query parameters and headers.

```
curl --location --request DELETE '$AI_API_URL/v2/lm/metrics?  
executionId=e1c49497ccf6dde8' \  
--header 'AI-Resource-Group: default' \  
--header 'Authorization: Bearer $TOKEN'
```

# 9 APIs and API Extensions

Explore APIs and API extensions that you can use with SAP AI Core.

APIs

Resource	Description	More Information
AI Core API	Explore the entities, methods, and endpoints provided by the API for SAP AI Core.	<a href="#">AI Core API</a> 
AI API	Explore the entities, methods, and endpoints provided by the runtime-agnostic AI API.	<a href="#">AI API swagger specification</a>
Prompt Registry	Simplify the lifecycle management of prompt templates of your business AI scenarios.	<a href="#">Prompt Registry API</a> 
Orchestration	Enhance content generation with additional capabilities for business AI scenarios.	<a href="#">Orchestration API</a> 

API Extensions

Resource	Description	More Information
Dataset Management Extension	Manage your dataset files.	<a href="#">AI API Dataset Extension swagger specification</a> .
Resource Groups Extension	Access usage information for a tenant using the AI API.	<a href="#">AI API Admin API swagger specification</a>
Analytics Extension	Add analytics capabilities to the AI API.	<a href="#">AI API Analytics Extension swagger specification</a>
Metrics Extension	Query which capabilities of the metrics endpoint are supported.	<a href="#">AI API Metrics API swagger specification</a>

# 10 Libraries and SDKs

Explore additional SDKs and libraries that you can use with SAP AI Core.

SDKs Available with SAP AI Core

Resource	Description	More Information
 SAP Cloud SDK for AI : JavaScript	<p>SAP Cloud SDK for AI is the official SDK for SAP AI Core, generative AI hub, and orchestration.</p> <p>You can use SAP Cloud SDK for AI to:</p> <ul style="list-style-type: none"><li>• Integrate chat completion into your business applications with SAP Cloud SDK for AI.</li><li>• Leverage the generative AI hub capabilities of SAP AI Core such as templating, grounding, data masking, content filtering. For more information, see <a href="#">SAP AI Core</a>.</li><li>• Setup your SAP AI Core instance with SAP Cloud SDK for AI.</li></ul>	<ul style="list-style-type: none"><li>• <a href="#">GitHub Repository</a> </li><li>• <a href="#">Documentation</a> </li><li>• <a href="#">NPM</a> </li></ul>
 SAP Cloud SDK for AI : Java	<p>SAP Cloud SDK for AI is the official SDK for SAP AI Core, generative AI hub, and orchestration.</p> <p>You can use SAP Cloud SDK for AI to:</p> <ul style="list-style-type: none"><li>• Integrate chat completion into your business applications with SAP Cloud SDK for AI.</li><li>• Leverage the generative AI hub capabilities of SAP AI Core such as templating, grounding, data masking, content filtering. For more information, see <a href="#">SAP AI Core</a>.</li><li>• Setup your SAP AI Core instance with SAP Cloud SDK for AI.</li></ul>	<ul style="list-style-type: none"><li>• <a href="#">GitHub Repository</a> </li><li>• <a href="#">Documentation</a> </li><li>• <a href="#">Maven</a> </li></ul>

Resource	Description	More Information
 SAP Cloud SDK for AI : Python	<p>SAP Cloud SDK for AI is the official SDK for SAP AI Core, generative AI hub, and orchestration. The SDK is composed of three python distributions:</p> <p>You can use sap-ai-sdk-base to access the AI API using Python methods and data structures.</p> <p>You can use sap-ai-sdk-core to interact with SAP AI Core for administration and public lifecycle management.</p> <p>You can use sap-ai-sdk-gen to:</p> <ul style="list-style-type: none"> <li>• Integrate native SDK libraries and langchain for accessing models on generative AI hub in SAP AI Core..</li> <li>• Leverage the orchestration service of generative AI hub with capabilities such as templating, grounding, data masking, and content filtering. For more information, see <a href="#">Generative AI Hub</a>.</li> </ul>	<ul style="list-style-type: none"> <li>• <a href="#">Pypi Base</a></li> <li>• <a href="#">Pypi Core</a></li> <li>• <a href="#">Pypi Gen</a></li> <li>• <a href="#">Documentation</a></li> </ul>

# 11 Content Packages

Explore additional Content Packages for use with SAP AI Core.

Content Packages Available with SAP AI Core

Resource	Description	More Information
Data robot package	The content package for DataRobot integration for SAP AI Core.	<a href="#">Data robot package for SAP AI Core</a> 
Computer vision package	The content package for image use cases in SAP AI Core adds image classification and feature extraction and is used with the SAP AI SDK Core.	<a href="#">Computer vision package for SAP AI Core</a> 

# 12 Tutorials

All available missions for SAP AI Core.

AI Use Case	Tutorial Group/Mission	Description
Generative AI	<a href="#">Generative AI with SAP AI Core - Setup</a>	Set up your SAP Business Technology Platform environment to explore SAP AI Core.
Generative AI	<a href="#">Generative AI with SAP AI Core - Orchestration</a>	Get started with generative AI workflows with different vendors of Large language models in SAP AI Core and learn the fundamentals of promoting and embedding with generative AI SDK in SAP AI Core.
Generative AI	<a href="#">Generative AI with SAP AI Core - Foundation Models</a>	Explore foundation large language models that are part of SAP AI Core in a number of use cases.
Predictive AI	<a href="#">Predictive AI with SAP AI Core</a>	Get started with SAP AI Core, learn the fundamentals, create your first predictive AI workflow, and move your machine learning code to a production cloud.

# 13 Advanced Features

Explore advanced features, within SAP AI Core.

## [AI Content as a Service \[page 577\]](#)

With SAP AI Core, you can publish AI content such as workflows, serving templates, or Docker images as a managed service on the SAP BTP *Service Marketplace*. This allows other tenants to consume your content through standard APIs.

## 13.1 AI Content as a Service

With SAP AI Core, you can publish AI content such as workflows, serving templates, or Docker images as a managed service on the SAP BTP *Service Marketplace*. This allows other tenants to consume your content through standard APIs.

A **service provider** is the main SAP AI Core tenant that publishes AI content as a service. For example, the service provider provides a workflow template for other users.

### Note

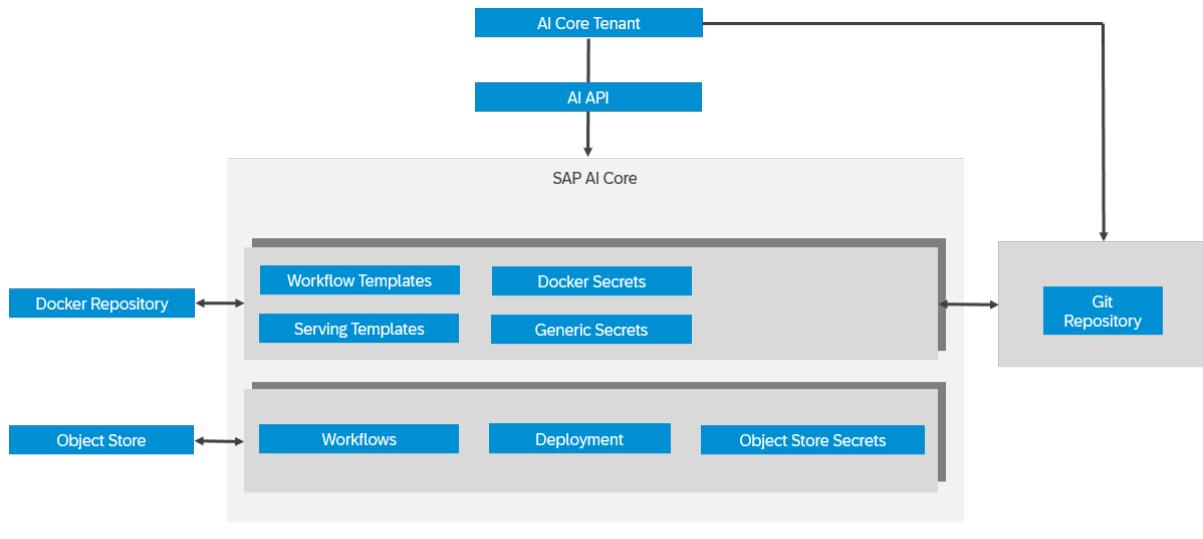
A service provider tenant can provision only one service.

A **service consumer** is the tenant that consumes AI content published by a provider. Consumers create service instances, generate service keys, and use the provided service URL to access the content. They can also start executions or deployments.

## Service Provider Flow

As a service provider, you publish your AI content to the SAP BTP *Service Marketplace* as follows:

1. Create consumer-ready AI content (for example, a workflow, serving template, or Docker image).
2. Create a generic secret for broker registration. For more information, see [Create a Generic Secret \[page 111\]](#).
3. Provide a service custom resource YAML in a registered git repository.
4. Fetch the service broker information by calling the endpoint: `{apiurl}/v2/admin/services`.
5. Register the service broker in the SAP BTP *Service Marketplace* and SAP Cloud Management service.

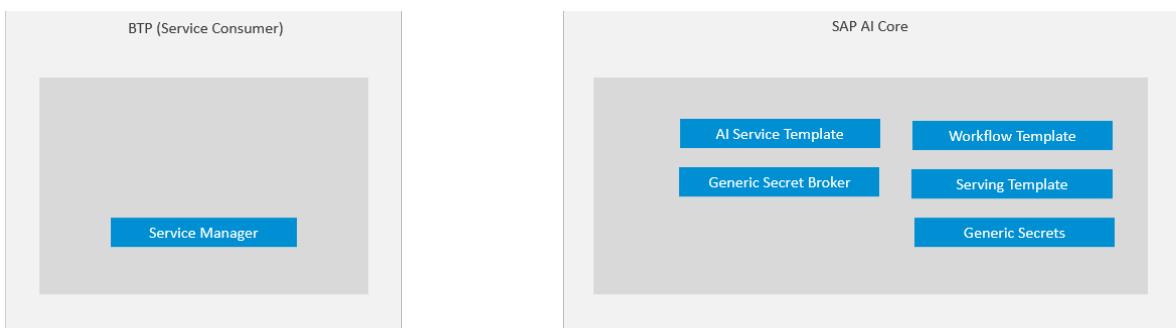


Your AI content is now available in the SAP BTP [Service Marketplace](#) for consumers to use. The service broker manages onboarding and offboarding of consumers automatically.

## Service Consumer Flow

As a service consumer, you create and use a service instance based on the provider's content.

1. In the SAP BTP [Service Marketplace](#), create a service instance.  
SAP AI Core creates a resource group for you with:  
`<resourceGroupId> == serviceInstanceId`.
2. Create a service key to authenticate against the service.
3. Use the service URL to start executions or deployments with the AI API (`serviceUrl`).



**Parent topic:** Advanced Features [page 577]

### 13.1.1 Service Custom Resource

The service provider main tenant needs to prepare the service custom resource. The custom resource contains service details, reference to broker credentials or secrets, and capabilities configured for service consumers.

An example service custom resource is provided in the following code block:

```
apiVersion: ai.sap.com/v1alpha1
kind: Service
metadata:
  name: sample-service
spec:
  brokerSecret:
    name: broker-credentials
    usernameKeyRef: username
    passwordKeyRef: password
  description: Service used for demos
  capabilities:
    basic:
      staticDeployments: true
      userDeployments: true
      createExecutions: true
      userPromptTemplates: true
  logs:
    executions: true
    deployments: true
  serviceCatalog:
  - extendCredentials:
    shared:
      serviceUrls:
        AI_API_URL: https://api.ai.internalprod.eu-
central-1.aws.ml.hana.ondemand.com
    extendCatalog:
      name: sample-service
      id: sample-service-broker-id
      description: sample service
      bindable: true
      plans:
        - id: sample-service-standard
          description: Standard plan for sample service
          name: standard
          free: false
          metadata:
            supportedPlatforms:
              - cloudfoundry
              - kubernetes
              - sapbtp
```

This can be used as a guide, with values amended as required. To create YAML descriptors, use any text editor with a YAML plugin.

For details about parameters, refer to the following table:

#### Service Parameter Details

Type	Parameter	Description	
metadata	name	Name for the service	
brokerSecret	name	Secret's name containing credentials to register Service Broker.	
		<p><b>① Note</b></p> <p>It is mandatory to have this secret registered as a generic secret.</p>	
	usernameKeyRef	Key reference for username from registered Secret.	
	passwordKeyRef	Key reference for password from registered Secret.	
capabilities	basic	staticDeployments	
		userDeployments	Consumers can use existing deployments, but not allowed to Create, Update or Delete (default: true).
		createExecutions	Consumers can create executions (default: true).
		userPromptTemplates	Consumers can create prompt templates (default: true).
logs	executions	Consumers can access execution logs.	
	deployments	Consumers can access deployment logs.	
	by default	Consumer has access to: <ul style="list-style-type: none"> <li>• Create, read artifacts &amp; configurations</li> <li>• Download, upload and delete datasets</li> <li>• Create, read, update &amp; delete object store secrets</li> </ul>	
serviceCatalog	extendCredentials	Used to mention to be consumed Service URL. It will reflect in the service-key.	
	extendCatalog	Used to extend service catalog.	

Type	Parameter	Description
	enableSharedResourceGroup	Flag used to enable or disable share resource group. For more information, see <a href="#">Shared Resource Group [page 584]</a> .

#### ⚠ Restriction

Updating of Service Custom Resource is not Supported

Once a service custom resource has synced successfully, updates of any parameters in templates will not take any effect.

## 13.1.2 Getting Started as a Service Provider

To onboard a service, complete the following:

1. Create a brokerSecret to be used as credentials when registering the service broker.
  1. Use a new POST request to URL `{ {apiurl} }/v2/admin/secrets`
  2. Provide credentials (base64 encoded) in the request body in JSON format:

```
{
  "name": "broker-credentials",
  "data": {
    "username": "bXktc2VjcmV0LWNyZWR1bnRpYWw=",
    "password": "bXktc2VjcmV0LW90aGVyLWNyZWR1bnRpYWw="
  }
}
```

3. Specify the scope of the request in the header: `AI-Tenant-Scope: true`
4. Send the request.

2. Modify the brokerSecret specification section using these details:

```
apiVersion: ai.sap.com/v1alpha1
kind: Service
metadata:
  name: sample-service
spec:
  brokerSecret:
    name: broker-credentials
    usernameKeyRef: username
    passwordKeyRef: password
  description: Service used for demos
  capabilities:
    basic:
      staticDeployments: true
      userDeployments: true
      createExecutions: true
    logs:
      executions: true
      deployments: true
  serviceCatalog:
    - extendCredentials:
        shared:
```

```

        serviceUrls:
          AI_SVC_URL: https://api.ai.internalprod.eu-
central-1.aws.ml.hana.ondemand.com
      extendCatalog:
        name: sample-service
        id: sample-service-broker-id
        description: sample service
        bindable: true
        plans:
          - id: sample-service-standard
            description: Standard plan for sample Service
            name: standard
            free: false
            metadata:
              supportedPlatforms:
                - cloudfoundry
                - kubernetes
                - sapbtp

```

### ⓘ Note

The username and password are key names from the broker credentials in the previous step.

3. Update the spec.serviceCatalog[].extendCredentials with the service URL you want to provide to the consumer, which will be part of the service key. Provide catalog details under spec.serviceCatalog[].extendCatalog.
4. Push your service custom resource to your registered GitHub repository, and wait for the sync to be successful.
5. Once it has synced, Check the service details by sending a GET request to URL { {apiurl} } /v2/admin/services.

```
{
  "count": 1
  "resources": [
    {
      "name": "sample-service",
      "description": "Service used for demos",
      "status": "PROVISIONED",
      "url": "https://aif-xyzabc.servicebroker.internalprod.eu-
central.aws.ml.hana.ondemand.com"
    }
  ]
}
```

Take note of the service broker URL.

6. Register the service broker using smctl as subaccount-scoped.
  1. Test the registration of the service broker first as subaccount-scoped, before you register it globally. Subaccount-scoped means that your service is automatically visible in the catalog of environments where it's registered. You can follow steps described in Service Manager Guide to [Set Up for smctl](#). Once smctl is installed, login as shown:

```
# env variables
SERVICE_MANAGER_URL=<sm url e.g. https://service-
manager.cfapps.sap.hana.ondemand.com/>
SVC_SUBACCOUNT_USER=<user-with-servicemanager-
role>SVC_SUBACCOUNT_PWD=<password + 2FA>
SERVICE_BROKER_URL=https://aif-xyzabc.servicebroker.internalprod.eu-
central.aws.ml.hana.ondemand.com
SVC_SUBACCOUNT_SUBDOMAIN=<subaccount e.g. subaccountxyz>
SERVICE_BROKER_USER=<broker username provided in secret>
SERVICE_BROKER_PWD=<broker password provided in secret>
```

```
# smctl login
smctl login -a $SERVICE_MANAGER_URL \
--param subdomain=$SVC_SUBACCOUNT_SUBDOMAIN \
-u=$SVC_SUBACCOUNT_USER \
-p=$SVC_SUBACCOUNT_PWD
```

2. Register your service-broker by providing the broker-name, URL, and credentials.

```
# register service-broker
smctl register-broker sample-service $SERVICE_BROKER_URL -b
$SERVICE_BROKER_USER:$SERVICE_BROKER_PWD
# service-broker registration should complete successfully
```

With the service-broker registered successfully, the service is available in the [Service Marketplace](#).

```
# assuming you are logged in to Cloud
Foundry, provided correct subaccount, get
service plan information
cf marketplace -s sample-service
```

Consumers can now create a service instance and service-key. On creation of service instance, SAP AI Core will create a corresponding resource-group with id = instance id, and the service is now ready for use.

### 13.1.3 Metering

Describes how SubaccountId and ServiceInstanceId are available as environment variables in the workflow runtime for metering.

For metering purposes, the workflow runtime automatically injects the following environment variables into each workflow pod:

- AICORE\_CAS\_SUBACCOUNT\_ID: The Subaccount ID of the CaaS consumer.
- AICORE\_CAS\_SERVICE\_INSTANCE\_ID: The Service Instance ID of the CaaS consumer.

These environment variables are available to all processes running inside the workflow pod. You can use them to track usage, or integrate with metering system.

In your workflow container, you can access these variables using the following bash script:

```
echo "Subaccount ID: $AICORE_SUBACCOUNT_ID"
echo "Service Instance ID: $AICORE_SERVICE_INSTANCE_ID"
```

### 13.1.4 Offboarding

To prevent accidental deletion of the service, service providers must provide a deletion strategy as follows:

```
metadata:
  name: sample-service
  annotations:
    ai.sap.com/serviceDeletionStrategy: "delete"
```

With the serviceDeletionStrategy annotation, service providers can delete service custom resources from the git repository and proceed for offboarding. For successful service offboarding, all the consumer service instances should be deleted.

### 13.1.5 Shared Resource Group

#### Context

When a consumer creates a service instance of the custom resource, a resource group is automatically created with the ID of the service instance ID. A consumer's access to resources is restricted to this resource group.

A shared resource group can be used if a service provider wants to manage a central deployment and make it available to all of the service consumers

#### ⚠ Restriction

Shared resource groups only allow deployment creation for models available in the foundation-models scenario, in generative AI hub.

#### Procedure

1. To enable shared resource group, set enableSharedResourceGroup to true in the service custom resource.

#### ↔ Sample Code

```
...  
    enableSharedResourceGroup: true  
    serviceCatalog:  
    ...
```

Tenants cannot create, modify or delete a shared resource group using the API. Onboarding for shared resource groups is only possible through the service custom resource..

2. Check the status of your shared resource group by sending a GET request to the endpoint `{apiurl}/v2/admin/services/<service-name>`.

#### ↔ Sample Code

```
curl --location '$AI_API_URL/v2/admin/services/aisvc-spam-detection' \  
--header 'Authorization: Bearer $TOKEN'  
# API Response:  
{  
    "brokerSecret": {  
        "name": "broker-credentials",  
        "passwordKeyRef": "password",  
    },  
}
```

```

        "usernameKeyRef": "username"
    },
    "capabilities": {
        "basic": {
            "createExecutions": true,
            "staticDeployments": true,
            "userDeployments": true
        },
        "logs": {
            "deployments": true,
            "executions": true
        }
    },
    "description": "Service exposing AI Content for Spam Detection use-case",
    "name": "aisvc-spam-detection",
    "serviceCatalog": [
        {
            "extendCatalog": {
                "bindable": true,
                "description": "Service exposing AI Content for Spam Detection use-case",
                "id": "aisvc-spam-detection-0",
                "name": "aisvc-spam-detection",
                "plans": [
                    {
                        "description": "Standard plan for Spam Detection Service",
                        "free": false,
                        "id": "standard",
                        "metadata": {
                            "supportedPlatforms": [
                                "cloudfoundry",
                                "kubernetes",
                                "sapbtp"
                            ]
                        },
                        "name": "standard"
                    }
                ]
            }
        },
        "extendCredentials": {
            "shared": {
                "serviceUrls": {
                    "AI_API_URL": "https://api.ai.internalprod.eu-central-1.aws.ml.hana.ondemand.com"
                }
            }
        }
    ],
    "sharedResourceGroupStatus": {
        "id": "shared",
        "isEnabled": true,
        "state": "Provisioned"
    },
    "status": "PROVISIONED",
    "statusMessage": "",
    "url": "https://aisvc-425c28c5-aisvc-spam-detection.servicebroker.internalprod.eu-central-1.aws.ml.hana.ondemand.com"
}

```

## **Results**

If the `sharedResourceGroupStatus` is `Provisioned`, your shared resource group is provisioned successfully.

## **Next Steps**

To add resources to your shared resource group, include the header `AI-Resource-Group: shared`. Consumers accessing service must also use the header `AI-Resource-Group: shared` to access the shared resource group.

# 14 Security

Here, we'll explain some of the security aspects of SAP AI Core.

## 14.1 Security Features of Data, Data Flow, and Processes

The table below shows an overview of the data flow for SAP AI Core.

Overview of Data and Security Measures

Step	Description	Security Measure
1	Transmission control/communication security	Encrypted (HTTPS) communication. Data-in-transit is encrypted using state-of-the-art TLS settings.
2	Application data residing in persistence layer	Encrypted at rest with state-of-the-art encryption keys generated and maintained by SAP.
3	Application data residing in persistence layer	Backup and restore capabilities are implemented and tested regularly. Backups are stored in remote locations and backups are encrypted at rest with state-of-the-art encryption keys generated and maintained by SAP.
4	Access control and separation by purpose	Roles and scopes are available for implementing access control. Customer admin can use standard BTP security administration capabilities to assign roles to users to ensure "least privilege" and "segregation of duties".

## 14.2 Encryption in Transit

Communication with the service, including data upload and download, is encrypted using the transport layer security (TLS) protocol. SAP services support only the latest protocol versions (that is, TLS v1.2 and later) and strong cipher suites. Your systems must use the supported protocol versions and cipher suites to set up secure communication with the services. They must also validate the certificates against the services' domain names to avoid man-in-the-middle attacks.

## 14.3 Authentication and Administration

SAP AI Core uses the authentication mechanisms provided by SAP Authorization and Trust Management service (XSUAA). The credentials to authenticate against the XSUAA to access SAP AI Core are provided as part of the service key for the SAP AI Core service.

For information about SAP Authorization and Trust Management service (XSUAA) in SAP BTP, see [What Is the SAP Authorization and Trust Management Service?](#)

## 14.4 Docker Images

SAP AI Core supports tenant-specific Docker registries (registered via the administration APIs). Additional tenant workloads, such as for execution and deployments, can be created by referencing the Docker images from this Docker registry.

Docker images are cached on virtual machines. These cached Docker images cannot be accessed by other tenants and will not be accessed by SAP.

Cached Docker images are not deleted immediately upon tenant offboarding but are cleaned up as part of operational events such as cluster scaling-down behavior, maintenance, and upgrade of virtual machines.

With every service that you consume, there is a shared security responsibility between you and SAP. Because the creation of a Docker image is the responsibility of the tenant, we strongly recommend that you do not embed or hard-code personal data, sensitive data, or machine learning models inside your Docker images.

For security reasons, Docker containers in SAP AI Core are run as non-root only. For more information, see [Workflow Templates \[page 464\]](#).

## 14.5 AI Content Security

AI content covers workflow templates and serving templates, as well as the Docker images used in them. Docker images contain the machine-learning algorithms or code, along with the machine-learning libraries, frameworks, and other dependent packages. Ensure that you follow standard practices for developing secure software when working with AI content.

### ⓘ Note

Users of AI core are responsible of the content of their docker images and assume the risk of running compromised containers in the platform.

## Security Practices

Practice	Description
Threat Modeling	Hold threat modeling workshops to identify and assess security risks or threats in the AI content.
Static Code Scans	Use static code scan (SAST) tools to scan and analyze the code for vulnerabilities.
Open-Source Vulnerability Scan	Scan any open-source components used by the product for vulnerabilities, and patch any vulnerable OSS components found.
Open-Source Strategy	Develop an update strategy that defines when the open-source components used in a product or service are updated to the latest secure version.
Code Reviews	Perform peer code reviews of each code change. The reviewer should take a closer look at the code from a security perspective.
Malware Scanning	Scan for malware in the data uploaded for AI content.
Secure Code Protection	Support security assurance starting with source code through deployed service. For example, use Docker image digest and image sign verifications.
Docker Base Image Security	Use a secure, light base image for building the Docker images for the AI content. Ensure you use the latest available base image and remove unused components from the Docker image.

## 14.6 Kubernetes Security

We recommend that you enable the relevant and applicable Kubernetes security features on your workflow templates and serving templates. Ensure that you enable the appropriate Kubernetes features for your workloads.

### Restricted Pod Settings

To ensure proper security configurations for containers, certain security-related pod fields must adhere to specific values. There is a large number of pod fields, which change over time. Here, we list the most important fields and their required settings. You do not have to explicitly set these values on your own, however if you set these fields incorrectly, the pod will be rejected.

#### Setting for `PodSecurityContext`

Capability	Setting
Add	Must not be set
Drop	Must contain exactly one element: [ 'all' ]
RunAsUser	Must be set to 65534

Capability	Setting
RunAsGroup	Must be set to 65534
RunAsNonRoot	Must be set to <code>true</code> . The pod must run as a non-root user
AllowPrivilegeEscalation	Must be set to <code>false</code>

These settings are enforced internally. If your pod is rejected, it is likely due to one or more of these fields being set incorrectly. Please review your configuration and ensure compliance with the above requirements.

## Related Information

[Security Best Practices for Kubernetes Deployment](#) ↗

## 14.7 Configuration Data and Secrets

Workloads can access network resources other than object stores by using credentials at runtime. The ways of including this information at runtime have different standards when it comes to confidentiality.

- For **sensitive** information:  
SAP AI Core allows you to include secrets in the form of generic secrets. Generic secrets are created and managed by the REST APIs in SAP AI Core and consumed securely in a workload.
- For **non-sensitive** information:  
You can include non-sensitive parameters using configurations or labels.

ⓘ Note

These parameters may be returned in clear text (for example, in GET requests).

## 14.8 Output Encoding

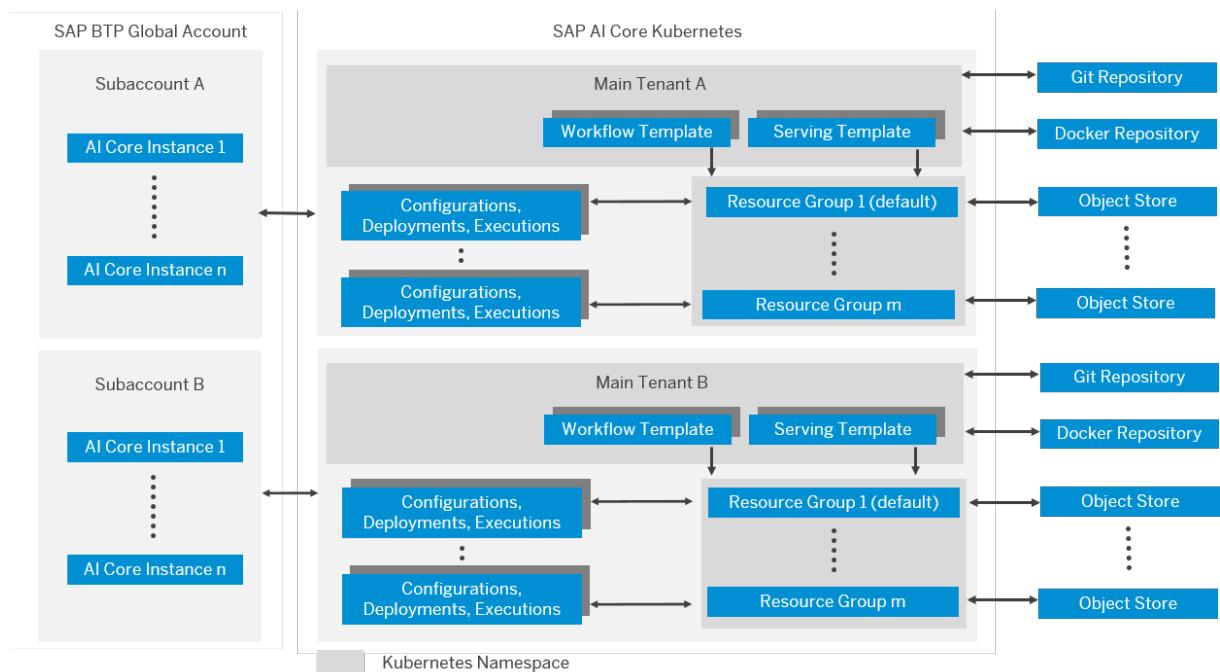
To avoid breaking the business functionality, SAP AI Core does not sanitize any user input. Consumers or applications that consume the AI API are expected to perform necessary output encoding based on the usage context to prevent XSS attacks.

## Related Information

[Cross Site Scripting \(XSS\) – OWASP Site](#) ↗

## 14.9 Multitenancy

SAP AI Core is a tenant-aware BTP reuse service, supporting main tenants and resource groups. Resources are defined for tenants or resource groups as outlined below:



Each main tenant and resource group is mapped to a namespace. The main tenant namespace only contains templates for workflows and model serving. The instances of these objects are created in the respective resource group namespaces and reference the corresponding templates in the main tenant namespace. Each main tenant has a default resource group, which can be used for workloads from the main tenant.

### Tenant-Level Resources

Tenant-level resources include executables such as:

- Workflow templates
- Serving templates
- Docker registries that contain Docker images
- User authentication and authorization (UAA)

User authentication and authorization are based on the SAP AI Core tenant (access token obtained using the service key for SAP AI Core). At runtime or when managing the lifecycle via AI API, the SAP AI Core tenant must set the appropriate resource group in the request header.

## 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.

## Tenant Isolation of Workloads

Workloads run in a sandbox environment and cannot access workflows of other tenants or resource groups. Only TCP is supported for inbound or outbound traffic from a workload. Opening workloads to open Sockets on UDP ports is strongly discouraged. They are not usable, but may pose a theoretical security problem for the workload.

## 14.10 Auditing and Logging Information

Here you can find a list of the security events that are logged by SAP AI Core.

Security Events Written in Audit Logs

What Events Are Logged	How to Identify Related Log Events
Creation of object store secret	Message containing a time stamp, tenant IDs, and attributes containing the API call and "new": "Succeeded".
Deletion of object store secret	Message containing a time stamp, tenant IDs, and attributes containing the API call and "new": "Succeeded".
Successful retrieval of object store secret	Successfully read object store connection details for resource group <resource group name> and object store <object store name>
Provisioning of resource group	Message containing a time stamp, tenant IDs, and attributes containing the API call and "new": "Succeeded".
Deprovisioning of resource group	Message containing a time stamp, tenant IDs, and attributes containing the API call and "new": "Succeeded".
Creation of docker registry secret	Message containing a time stamp, tenant IDs, and attributes containing the API call and "new": "Succeeded".

What Events Are Logged	How to Identify Related Log Events
Deletion of docker registry secret	Message containing a time stamp, tenant IDs, and attributes containing the API call and "new": "Succeeded".
Creation of deployments	Message containing a time stamp, tenant IDs, and attributes containing the API call and "new": "Succeeded".
Deletion of deployments	Message containing a time stamp, tenant IDs, and attributes containing the API call and "new": "Succeeded".
Creation of executions	Message containing a time stamp, tenant IDs, and attributes containing the API call and "new": "Succeeded".
Deletion of executions	Message containing a time stamp, tenant IDs, and attributes containing the API call and "new": "Succeeded".
Creation of ArgoCD application	Message containing a time stamp, tenant IDs, and attributes containing the API call and "new": "Succeeded".
Deletion of ArgoCD application	Message containing a time stamp, tenant IDs, and attributes containing the API call and "new": "Succeeded".
Creation of repositories	Message containing a time stamp, tenant IDs, and attributes containing the API call and "new": "Succeeded".
Deletion of repositories	Message containing a time stamp, tenant IDs, and attributes containing the API call and "new": "Succeeded".
Provisioning of tenant	Message containing a time stamp, tenant IDs, and attributes containing "name": "state",, "new": "PENDING" and "name": "cfAccountState, "new": "ACTIVE" and success value true
Retrieval of tenant	Message containing Attribute <attribute detail> was read.
Deprovisioning of tenant	Message containing a time stamp, tenant IDs, and attributes containing "name": "state",, "new": "PENDING" and "name": "cfAccountState, "new": "DELETED" and success value true.
List	Message containing a time stamp, tenant IDs, source IPs, the request URI, and the level.
Get	
Watch	
Create	Message containing a time stamp, tenant IDs, source IPs, the request URI, the level, and the request and response objects.
Update	
Patch	

What Events Are Logged	How to Identify Related Log Events
Delete	Message containing a time stamp, tenant IDs, source IPs, the request URI, the level, the and response object.
Token expired	<code>Jwt is expired</code>
No authorization header	<code>RBAC: access denied</code>
Invalid token	<code>Jwt issuer is not configured</code>
Wrong tenantId	<code>Jwt verification fails</code>

The following information is described in the table columns:

- *Event grouping* - Events that are logged with a similar format or are related to the same entities.
- *What events are logged* - Description of the security or data protection and privacy related event that is logged.
- *How to identify related log events* - Search criteria or key words, that are specific for a log event that is created along with the logged event.
- *Additional information* - Any related information that can be helpful.

## Related Information

[Audit Logging in the Cloud Foundry Environment](#)

[Audit Logging in the Neo Environment](#)

## 14.11 Data Protection and Privacy

For general information about data protection and privacy on SAP Business Technology Platform, see [Data Protection and Privacy](#).

Data protection is associated with numerous legal requirements and privacy concerns. In addition to compliance with general data protection and privacy acts, it is necessary to consider compliance with industry-specific legislation in different countries/regions. This section describes the specific features and functions that SAP AI Core provides to support compliance with the relevant legal requirements and data privacy.

This guide does not give advice on whether these features and functions are the best method to support company, industry, regional, or country/region-specific requirements. Furthermore, this guide does not give advice or recommendations about additional features that would be required in a particular environment. Decisions related to data protection must be made on a case-by-case basis and under consideration of the given system landscape and the applicable legal requirements.

### Note

SAP does not provide legal advice in any form. SAP software supports data protection compliance by providing security features and specific data protection-relevant functions, such as simplified blocking and

deletion of personal data. In many cases, compliance with applicable data protection and privacy laws will not be covered by a product feature. Definitions and other terms used in this document are not taken from a particular legal source.

The extent to which data protection is ensured depends on secure system operation. Network security, security note implementation, adequate logging of system changes, and appropriate usage of the system are the basic technical requirements for compliance with data privacy legislation and other legislation.

For a glossary of Data Protection and Privacy terms in SAP BTP, see the SAP BTP [Glossary for Data Protection and Privacy](#).

### 14.11.1 Data Storage and Processing

SAP AI Core provides functionality that allows you to process data, such as configuration files or Machine Learning (ML) Training or ML Serving.

SAP AI Core acts as the data processor and is not aware of the type of data or category of data. SAP AI Core customers, as Data Controllers, are responsible for fulfilling data protection and privacy (DPP) responsibilities for data storage and processing requirements.

### 14.11.2 Change Logging and Read-Access Logging

SAP AI Core acts as the data processor and is not aware of the type of data or category of data. SAP AI Core customers, as Data Controllers, are responsible for fulfilling data protection and privacy (DPP) responsibilities for data storage and processing requirements.

For any applications or services you develop using SAP AI Core, you must ensure that they include relevant logging functions, and ensure compliance with the data privacy laws by making sure that the data is properly logged.

### 14.11.3 Consent

SAP AI Core acts as the data processor and is not aware of the type of data or category of data. SAP AI Core customers, as Data Controllers, are responsible for fulfilling asking for consent from data subjects before collecting any personal data.

## 14.11.4 Deletion

SAP AI Core supports **Bring your object store**, whereby customers register the object store secret where artifact relevant files (such as training data or machine learning models or other types) are stored. Such data is used by the ML workloads during processing, such as ML Training or ML Serving.

The notion of artifacts is limited to capture the metadata of the data. Data is physically stored in the object store and SAP AI Core is not responsible for deletion of files. When artifacts are deleted using AI API, corresponding metadata are deleted from SAP AI Core service and the actual files are not deleted from the object store. For more information about the AI API, see [AI API Overview \[page 50\]](#).

Upon offboarding, SAP AI Core will clean up the cached data within AI Core used for processing purposes.

SAP AI Core customers, as Data Controllers, are responsible for deletion of data from the registered object store.

## 14.11.5 Security and Customer Data Protection

SAP product standard security and the data protection and privacy (DPP) requirements set high standards and obligations when it comes to securing and protecting customer data that is entrusted to SAP.

Customer data protection is handled in three ways:

- Customer data is imported, output, and processed by the services for no purpose beyond that to which the customer has subscribed.
- Customer data is protected from malicious access by security technologies that include authentication and authorization.
- Customer data is protected from accidental exposure to SAP administrators or support persons by security policies, access controls, and monitoring.

# 15 Accessibility Features in SAP AI Core

To optimize your experience of SAP AI Core, SAP AI Core provides features and settings that help you use the software efficiently.

## Note

SAP AI Core uses SAP AI Launchpad for its interface, which is based on SAPUI5. For this reason, accessibility features for SAPUI5 also apply. See the accessibility documentation for SAPUI5 on SAP Help Portal at [Accessibility for End Users](#).

For more information on-screen reader support and keyboard shortcuts, see [Screen-Reader Support for SAPUI5 Controls](#) and [Keyboard Handling for SAPUI5 Elements](#).

# 16 Monitoring and Troubleshooting

Explore solutions to potential issues, and find out how to get support.

## Getting Support

If you encounter an issue with this service, we recommend that you follow the procedure below.

### Check Platform Status

Check the availability of the platform at [SAP Trust Center](#).

For more information about platform availability, updates and notifications, see [Platform Updates and Notifications in the Cloud Foundry Environment](#).

### Check Guided Answers

In the SAP Support Portal, check the [Guided Answers](#) section for SAP Business Technology Platform. You can find solutions for general SAP Cloud Platform issues as well as for specific services there.

### Contact SAP Support

You can report an incident or error through the [SAP Support Portal](#).

Please use the following component(s) for your incident:

Component Name	Component Description
CA-ML-AIC	SAP AI Core

We recommend that you include the following information when you submit the incident:

- Region information
- Subaccount technical name
- URL of the page where the incident or error occurs
- Steps or clicks used to replicate the error
- Screenshots, videos, or the code entered

## 16.1 Troubleshooting

For troubleshooting information, see the following sections:

[Repository \[page 599\]](#)

[Configuration \[page 600\]](#)

[Artifacts \[page 602\]](#)

[Application \[page 604\]](#)

[Execution \[page 609\]](#)

[Docker \[page 611\]](#)

[Deployment \[page 612\]](#)

[Miscellaneous \[page 613\]](#)

## 16.1.1 Repository

### Repository ra-aicore-test not found for tenant

You get the result:

```
{  
  "error": {  
    "code": "500",  
    "details": [  
      {  
        "code": null,  
        "message": "Repository ra-aicore-test not found for tenant  
b82a8318"  
      }  
    ],  
    "message": "Repository ra-aicore-test not found for tenant b82a8318",  
    "request_id": null,  
    "target": "/api/v4alpha/repositories"  
  }  
}
```

and:

```
AIAPIServerException: Failed to post /admin/repositories: Repository ra-aicore-  
test not found for tenant 68
```

#### Follow the solution:

Use a different name for the value of the name parameter. The exception is raised by reuse of the name aicore-test.

```
response = ai_api_client.rest_clinet.post(  
  path="/admin/repositories",  
  body={  
    "name": "aicore-test-1",  
    "url": "https://github.com/John/aicore-test",  
  }  
)  
print(response)  
{'message': 'Repository has been on-boarded.'
```

## Related Information

[Configuration \[page 600\]](#)

[Artifacts \[page 602\]](#)

[Application \[page 604\]](#)

[Execution \[page 609\]](#)

[Docker \[page 611\]](#)

[Deployment \[page 612\]](#)

[Miscellaneous \[page 613\]](#)

### 16.1.2 Configuration

#### Could not create configuration, because executable <*x*> for scenario <*y*> is not found

When you try to create a configuration, you are told that an executable cannot be found for your scenario.

##### Check the following:

1. Check that you are using the `name` value from your workflow for the executable ID in the configuration.

```
apiVersion: argoproj.io/v1alpha1
kind: WorkflowTemplate
metadata:
  name: text-clf-train-tutorial
  annotations:
    scenarios.ai.sap.com/description: "SAP developers tutorial
scenario"
  ...
  
```

##### ⓘ Note

Do not use the value from `executables.ai.sap.com/id` as an executable ID.

2. Check that you are using the value of `executables.ai.sap.com/id` from your workflows as your scenario ID.

```
...
  artifacts.ai.sap.com/text-data.kind: "dataset"
  artifacts.ai.sap.com/text-model-tutorial.kind: "model"
  labels:
    scenarios.ai.sap.com/id: "text-clf-tutorial"
    ai.sap.com/version: "2.1.0"
  spec:
    ...
    
```

...

## Log message: using minio client

### Check the following:

1. Check that you are using the name value from your workflow for the executable ID in the configuration.

```
apiVersion: argoproj.io/v1alpha1
kind: WorkflowTemplate
metadata:
  name: text-clf-train-tutorial
  annotations:
    scenarios.ai.sap.com/description: "SAP developers tutorial
scenario"
  ...
  
```

#### ⓘ Note

Do not use the value from executables.ai.sap.com/id as an executable ID.

2. Check that you are using the value of executables.ai.sap.com/id from your workflows as your scenario ID.

```
...
  ...
  artifacts.ai.sap.com/text-data.kind: "dataset"
  artifacts.ai.sap.com/text-model-tutorial.kind: "model"
  labels:
    scenarios.ai.sap.com/id: "text-clf-tutorial"
    ai.sap.com/version: "2.1.0"
  spec:
  ...
  
```

[Parent topic: Troubleshooting \[page 598\]](#)

## Related Information

[Repository \[page 599\]](#)

[Artifacts \[page 602\]](#)

[Application \[page 604\]](#)

[Execution \[page 609\]](#)

[Docker \[page 611\]](#)

[Deployment \[page 612\]](#)

[Miscellaneous \[page 613\]](#)

### 16.1.3 Artifacts

#### No output artifact has been generated

##### Complete the following:

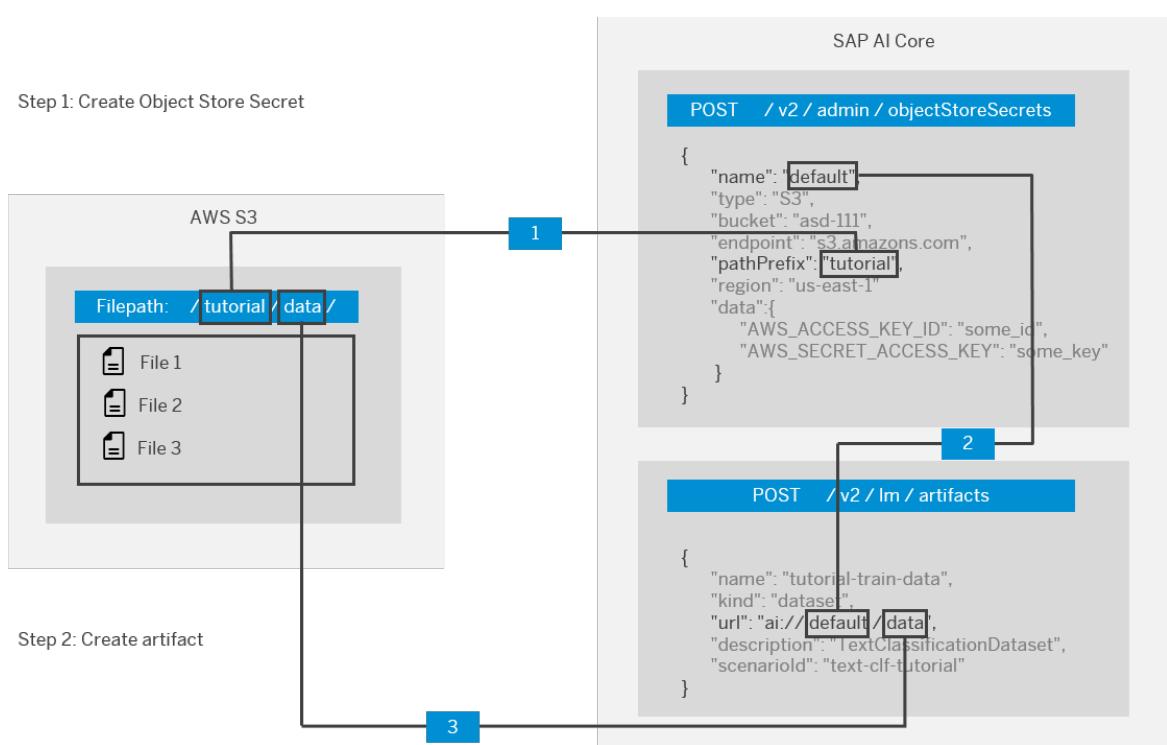
Define the globalName parameter for the output artifact in your workflow:

```
...  
    executables.ai.sap.com/description: "Text classification Scikit training  
executable"  
    executables.ai.sap.com/name: "text-clf-train-tutorial-exec"  
    artifacts.ai.sap.com/text-data.kind: "dataset"  
    artifacts.ai.sap.com/text-model-tutorial.kind: "mind"  
labels:  
    scenarios.ai.sap.com/id: "text-clf-tutorial"  
    ai.sap.com/version: "2.1.0"  
...  
...  
outputs:  
    artifacts:  
        -name: text-model-tutorial  
        path: /app/model  
        global name: text-model-tutorial  
        archive:  
            none: {}  
    container:  
...  
...
```

#### Failed to load artifacts: The specified key does not exist

##### Complete the following:

1. Ensure you have created an object store secret using the naming convention <name> and the pathPrefix from your AWS S3 path. Refer to the following diagram:



2. When creating the artifact, don't add the trailing forward slash (/) in URL parameter:

- Incorrect usage: `"url": "ai://yourObjectStoreSecretName/folder/subfolder/"`
- Correct usage: `"url": "ai://yourObjectStoreSecretName/folder/subfolder"`

**Parent topic:** Troubleshooting [page 598]

## Related Information

[Repository \[page 599\]](#)

[Configuration \[page 600\]](#)

[Application \[page 604\]](#)

[Execution \[page 609\]](#)

[Docker \[page 611\]](#)

[Deployment \[page 612\]](#)

[Miscellaneous \[page 613\]](#)

## 16.1.4 Application

### Templates are not synced via applications

After you have deleted or created an application, templates are not synced.

#### Follow the solution:

1. Delete the synced application using the endpoint:

```
DELETE {{apiurl}}/v2/admin/applications/{{appName}}
```

2. Offboard the associated GitHub repository using the endpoint:

```
DELETE {{apiurl}}/v2/admin/repositories/{{repositoryName}}
```

3. Onboard the associated GitHub repository using a personal access token instead of your GitHub password.

For more information, see [Creating a personal access token](#).

```
POST {{apiurl}}/v2/admin/repositories
```

#### ↔ Sample Code

Body:

```
{
  "name": "aicore-test",
  "url": "https://github.com/john/aicore-test",
  "username": "john",
  "password": "yourGitHubPersonalAccessToken"
}
```

4. Create the application using the endpoint:

```
POST {{apiurl}}/v2/admin/applications
```

5. Check the ArgoCD applications to determine if the repository has been synchronized correctly for the tenant. For example, check that there are no duplicated workflow names. The value of the name parameter is considered as an executable ID.

```
name: text-clf-train-tutorial
  annotations:
    ...apiVersion: argoproj.io/v1alpha1
kind: WorkflowTemplate
metadata:
```

6. Check that you're calling SAP AI Core using the expected tenant.

7. Check if the workflow templates contain the correct scenario label.

8. Get your application sync status using the endpoint:

```
GET {{apiurl}}/v2/admin/applications/{{appName}}/status
```

The status will return errors in your templates. When your templates are updated, the application will resync automatically after approximately three minutes.

## Executables do not appear when you retrieve them from a scenario

### Follow the solution:

1. Delete the synced application using the endpoint:

```
DELETE {{apiurl}}/v2/admin/repositories/{{appName}}
```

2. Offboard the associated GitHub repository using the endpoint:

```
DELETE {{apiurl}}/v2/admin/repositories/{{repositoryName}}
```

3. Onboard the associated GitHub repository using a personal access token instead of your GitHub password.

For more information, see [Creating a personal access token](#).

```
POST {{apiurl}}/v2/admin/repositories
```

#### ↔ Sample Code

Body:

```
apiVersion: argoproj.io/v1alpha1
"name": "aicore-test",
"url": "https://github.com/john/aicore-test",
"username": "john",
"password": "yourGitHubPersonalAccessToken"
}
```

4. Create the application using the endpoint:

```
POST {{apiurl}}/v2/admin/applications
```

5. Check the ArgoCD applications to determine if the repository has been synchronized correctly for the tenant. For example, check that there are no duplicated workflow names. The value of the name parameter is considered as an executable ID.

```
apiVersion: argoproj.io/v1alpha1
kind: WorkflowTemplate
metadata:
  name: text-clf-train-tutorial
  annotations:
    ...

```

6. Check if the user is calling SAP AI Core using the expected tenant.

7. Check that the scenario label is correct in the workflow templates.

8. Get your application sync status using the endpoint:

```
GET {{apiurl}}/v2/admin/applications/{{appName}}/status
```

The status will return errors in your templates. When your templates are updated, the application will resync automatically after approximately three minutes.

## Application status returns `healthy`, but most other properties are `unknown`

### Follow the solution:

1. Delete the synced application using the endpoint:

```
DELETE {{apiurl}}/v2/admin/applications/{{appName}}
```

2. Offboard the associated GitHub repository using the endpoint:

```
DELETE
```

```
 {{apiurl}}/v2/admin/repositories/{{repositoryName}}
```

3. Onboard the associated GitHub repository using a personal access token instead of your GitHub password.

For more information, see [Creating a personal access token](#).

**POST** {{apiurl}}/v2/admin/repositories

#### ↔ Sample Code

Body:

```
{  
  "name": "aicore-test",  
  "url": "https://github.com/john/aicore-test",  
  "username": "john",  
  "password": "yourGitHubPersonalAccessTokenHere"  
}
```

4. Create the application using the endpoint:

**POST** {{apiurl}}/v2/admin/applications

5. Check the ArgoCD applications to determine if the repository has been synchronized correctly for the tenant. For example, check that there are no duplicated workflow names. The value of the `name` parameter is considered as an executable ID.

```
apiVersion: argoproj.io/v1alpha1  
kind: WorkflowTemplate  
metadata:  
  name: text-clf-train-tutorial  
  annotations:  
    ...
```

6. Check if the user is calling SAP AI Core using the expected tenant.

7. Check if the workflow templates contain the correct scenario label.

8. Get your application sync status using the endpoint:

**GET** {{apiurl}}/v2/admin/applications/{{appName}}/status

The status will return errors in your templates. When your templates are updated, the application will resync automatically after approximately three minutes.

**Application status message:** rpc error: code = Unknown desc = my-path: app path does not exist

The specified path in your application doesn't exist in your repository.

**Follow the solution:**

Delete your application and create a new one using the correct path.

**Application status message:** application repo <your git repository> is not permitted in project 'xyz'

The repository URL cannot be found in your onboarded repositories.

### **Check the following:**

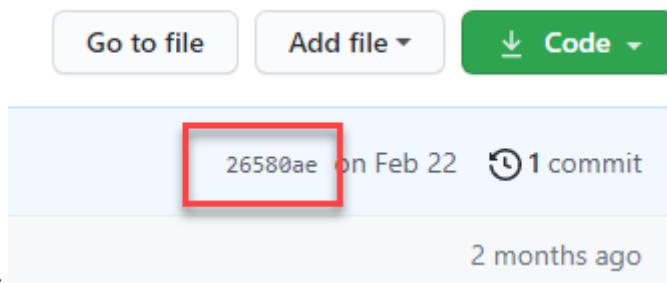
Make sure that the repository specified in your application has been successfully onboarded by using [GET](#) `{apiurl}/v2/admin/applications` and checking that the repository url has "status": "COMPLETED"

**Application status message:** `rpc error: code = Unknown desc = Unable to resolve 'not-existing-branch' to a commit SHA`

The revision you specified in your application doesn't exist in your repository.

### **Follow the solution:**

Delete your application and create a new one using the correct revision. The revision number is found in GitHub



Alternatively, enter [HEAD](#) to refer to the latest commit.

**Application status message:** `rpc error: code = FailedPrecondition desc = Failed to unmarshal \"workflow.yaml\": failed to unmarshal manifest: error converting YAML to JSON: yaml: line 7: mapping values are not allowed in this context`

You have a syntax error in your workflow template.

### **Follow the solution:**

Use Argo Lint to identify syntax errors in your workflow template. To set up the Argo Lint IDE, refer to [Argo Lint IDE setup](#) ↗

**Application status message:** `spec.source.repoURL and spec.source.path either spec.source.chart are required`

You specified an empty path in your application.

```
{  
  "healthStatus": "Unknown",
```

```
    "message": "spec.source.repoURL and spec.source.path either  
spec.source.chart are required",  
    "reconciledAt": "Unknown",  
    "source": {  
        "path": "Unknown",  
        "repoURL": "Unknown",  
        "revision": "Unknown"  
    },  
    "syncFinishedAt": "Unknown",  
    "syncRessourcesStatus": [ ],  
    "syncStartedAt": "Unknown",  
    "syncStatus": "Unknown"  
}
```

### Follow the solution:

Delete your application and create a new one, specifying a path. Check its status by using the endpoint:  
`{apiurl}/v2/admin/applications/{appName}/status`

## Sync an Application Manually

Applications sync with your GitHub repository automatically at intervals of ~3 minutes. Use the endpoint below to manually request a sync:`{apiurl}/admin/applications/{appName}/refresh`

**Parent topic:** [Troubleshooting \[page 598\]](#)

## Related Information

[Repository \[page 599\]](#)

[Configuration \[page 600\]](#)

[Artifacts \[page 602\]](#)

[Execution \[page 609\]](#)

[Docker \[page 611\]](#)

[Deployment \[page 612\]](#)

[Miscellaneous \[page 613\]](#)

## 16.1.5 Execution

### Execution status is DEAD or PENDING for a long time

#### Check the following:

1. Check the execution logs.
2. Check that the parameter name and execution name that you provided match those in your template. Note that the names are case-sensitive.

### Images from private docker.io cannot be pulled for execution

#### Follow the solution:

1. Specify the domain name of your Docker Hub. For example:

```
...
    none: {}
  container:
    image: "docker.io/tutorialrepo/movie-review-clf-
serve@sha256:123f4qwertyiop13432"
    imagePullPolicy: Always
...
```

If you do not already know your domain name, refer to the image:

The screenshot shows the Docker Hub interface. At the top, there's a search bar with 'Search for great content (e.g., mysql)' and navigation links for 'Explore' and 'Repositories'. Below the search bar, a breadcrumb navigation shows the path: 'tutorialrepo > Repositories > movie-review-clf-serve > latest'. The main content area displays the image details for 'tutorialrepo/movie-review-clf-serve:latest'. It features a blue hexagonal icon representing the image. To the right, the image name is shown as 'tutorialrepo/movie-review-clf-serve:latest' with a red box highlighting the digest 'DIGEST: sha256:123f4qwertyiop13432'. Below this, technical details are listed: 'OS/ARCH: linux/amd64', 'COMPRESSED SIZE: 2.7 GB', and 'LAST PUSHED: 16 days ago by tutorialrepo'.

2. Specify your Docker Image digest instead of the image version in your templates:  
image tutorialrepo/movie-review-clf-serve@sha256:123f4qwertyiop13432

## **Output artifacts in the execution list are empty and have status UNKNOWN for a long time**

### **Check the following:**

1. Check that an object secret with the name default exists in the current resource group. For more information, see [Register an Object Store Secret \[page 96\]](#)
2. Check that you have created the Docker Registry secrets required to pull your Docker Images. Logs will be available only after the execution has started.

## **GET execution has the status UNKNOWN for a long time**

### **Check the following:**

1. Check if an object secret with the name default exists in the current resource group. For more information, see [Register an Object Store Secret \[page 96\]](#)
2. Check that you have created the Docker Registry secrets required to pull your Docker Images. Logs will be available only after the execution has started.

## **Execution status changes to DEAD without any log**

### **Check the following:**

1. Check that your templates meet the Argo specifications and that they can be executed by SAP AI Core.
2. To identify errors automatically in your templates, use the Argo linter.
3. Check that you have created Docker folders, to store artifacts that will be created during an execution.

Parent topic: [Troubleshooting \[page 598\]](#)

## **Related Information**

[Repository \[page 599\]](#)

[Configuration \[page 600\]](#)

[Artifacts \[page 602\]](#)

[Application \[page 604\]](#)

[Docker \[page 611\]](#)

[Deployment \[page 612\]](#)

[Miscellaneous \[page 613\]](#)

## 16.1.6 Docker

### Error pulling container main logs. Back-off pulling image

You push a template to SAP AI Core and you get the error "Error pulling container main logs". "Back-off pulling image".

#### Complete the following steps:

1. Make sure that your Docker Registry is public-facing and not protected behind the firewall of your organization.  
If your Docker Image isn't public, verify that you have created a Docker Registry secret in SAP AI Core.
2. Check that you can pull your Docker Image on your local computer using the credentials from the previous step.
3. Specify the same Docker Registry secret in your executable.
4. Specify the path to your Docker Image in your executable in following format:  
`<DOCKER_REGISTRY>/<REPO_NAME>/<DOCKER_IMAGE>:<TAGNAME>`

#### Example

```
docker.io/tutorialrepo/text-clf-train:0.0.1
```

**Parent topic:** [Troubleshooting \[page 598\]](#)

## Related Information

[Repository \[page 599\]](#)

[Configuration \[page 600\]](#)

[Artifacts \[page 602\]](#)

[Application \[page 604\]](#)

[Execution \[page 609\]](#)

[Deployment \[page 612\]](#)

[Miscellaneous \[page 613\]](#)

## 16.1.7 Deployment

### You want to force a deployment with status UNKNOWN to stop

You want to stop and delete a deployment but you are unable to because the deployment status is “Unknown”. You have tried to submit a PATCH request as follows:

```
PATCH {{apiurl}}/lm/deployments/d4fec9c24c54f87e
```

However, you receive the following response:

```
{  
    "error": {  
        "code": "01010076"  
        "message": "Invalid Request, Current status UNKNOWN cannot be changed..",  
        "requestID": "e110820e-1cfe-456a-bb0e-77907b36422c",  
        "target": "/apu/v2/deployments/d4fec9c24c54f87e"  
    }  
}
```

#### Complete the following steps:

1. Find out why your deployment status is “Unknown” by using the endpoint:  
`GET {{apiurl}}/v2/lm/deployments/{{deploymentid}}`
2. Delete the deployment without trying to stop it (stopping a deployment is necessary only when it's running):  
`DELETE {{apiurl}}/v2/lm/deployments/{{deploymentid}}`

### Deployment remains in status PENDING

#### Check the following:

1. Check that the Docker Registry secret exists when using your private Docker Image.
2. Check that your Docker Image can be downloaded to your local system.

### Deployment ID <abc> not found

This message appears when you have just started the deployment. Wait a few minutes and the message will resolve itself automatically.

**Parent topic:** [Troubleshooting \[page 598\]](#)

## Related Information

[Repository \[page 599\]](#)

[Configuration \[page 600\]](#)

[Artifacts \[page 602\]](#)

[Application \[page 604\]](#)

[Execution \[page 609\]](#)

[Docker \[page 611\]](#)

[Miscellaneous \[page 613\]](#)

### 16.1.8 Miscellaneous

#### 403 - forbidden: RBAC Access denied

When you submit a POST request for an execution or a configuration, you get the error 403 – forbidden: RBAC Access denied.

##### Check the following:

1. Check that you are passing the correct token and AI-Resource-Group header.
2. Check your tenant provisioning.

#### You get a runtime adapter exception

You get the error: "Runtime Adapter Exception; Failed to post deployments : {\n\"code\": \"400\", \n\"message\": \"Missing input parameter or artifacts, one or more placeholder values are not resolved in the serving spec and error is 'dict object' has no attribute ' '.\\n\"\n"

##### Check the following:

Check that your input artifact key doesn't contain any separators, such as "example-artifact-key". If it does, rename the key (for example, "exampleArtifactKey")

## You have pushed your template to your GitHub repository but you can't see the executable created via the API

### There may be an error in your template. Check the following:

- Serving templates always expect an input parameter to be configured. If you don't have a parameter in your Serving template, add a dummy parameter and create a configuration for it.
- Hyphens (-) are the only separator as permitted in the template name.

## You have created a scenario in a template but you cannot see it in the AI API calls

### Use the following solution:

Add a workflow template with the new `scenario_id` (not a serving template) to make the scenario visible.

## Error: getaddrinfo ENOTFOUND

### Check the following:

1. Check that all environment variables match your SAP AI Core service keys. Specifically, check:
  - `auth_url`
  - `client_id`
  - `client_secret`
  - `apiurl`
2. Submit a GET request to the endpoint `{apiurl}/v2/admin/repositories`
3. If the issue persists, contact SAP support as described at [Monitoring and Troubleshooting \[page 598\]](#).

## Git repo doesn't synchronize with SAP AI Core instances.

When you try to synchronize your git repository with SAP AI Core, the response shows empty fields.

### Check the following:

1. Check that you are using a GitHub personal access token and not your GitHub password.

If you are already using a personal access token, proceed as follows:

1. Delete all UNKNOWN applications from your SAP AI Core instance using the endpoint:  
`DELETE {apiurl}/v2/admin/applications/{appName}`
2. Offboard your GitHub repo from SAP AI Core by calling the endpoint:  
`DELETE {apiurl}/v2/admin/repositories/{repositoryName}`
3. Generate a GitHub personal access token and use it to onboard your git repo to SAP AI Core as before. For more information, see [Creating a personal access token](#).

4. Sync your application again.

**Parent topic:** [Troubleshooting \[page 598\]](#)

## Related Information

[Repository \[page 599\]](#)

[Configuration \[page 600\]](#)

[Artifacts \[page 602\]](#)

[Application \[page 604\]](#)

[Execution \[page 609\]](#)

[Docker \[page 611\]](#)

[Deployment \[page 612\]](#)

# 17 Support Process

Explore solutions to potential issues, and find out how to get support.

## Getting Support

If you encounter an issue with this service, we recommend that you follow the procedure below.

### Check Platform Status

Check the availability of the platform at [SAP Trust Center](#).

For more information about platform availability, updates and notifications, see [Platform Updates and Notifications in the Cloud Foundry Environment](#).

### Check Guided Answers

In the SAP Support Portal, check the [Guided Answers](#) section for SAP Business Technology Platform. You can find solutions for general SAP Cloud Platform issues as well as for specific services there.

### Contact SAP Support

You can report an incident or error through the [SAP Support Portal](#).

Please use the following component(s) for your incident:

Component Name	Component Description
CA-ML-AIC	SAP AI Core

We recommend that you include the following information when you submit the incident:

- Region information
- Subaccount technical name
- URL of the page where the incident or error occurs
- Steps or clicks used to replicate the error
- Screenshots, videos, or the code entered

# 18 Service Offboarding

Tenant offboarding occurs when a customer deletes a subaccount. SAP AI Core polls for the subaccount deletion event and performs the necessary deprovisioning and deletion activities.

## ⓘ Note

Data and resources are not deleted when a service instance is deleted (because we don't isolate based on the service instance). If you want to keep the subaccount but still deprovision SAP AI Core, create a **medium support ticket** on component CA-ML-AIC with the title **Service Offboarding** and request that your data and resources be deleted manually.

# 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.



