/ Generative AI hub SDK

Generative AI hub SDK

With this SDK you can leverage the power of generative models available in the generative AI Hub of SAP AI Core. The generative AI Hub SDK provides model access by wrapping the native SDKs of the model providers (OpenAI, Amazon, Google), through langchain, or through the orchestration service.

Installation

To install this SDK, use the following pip command, which includes support for all models in GenAI Hub including langchain support:

```
pip install "generative-ai-hub-sdk[all]"
```

The default installation only includes OpenAI models (w/out langchain support):

```
pip install generative-ai-hub-sdk
```

You can install a subset of the extra libraries (w/out langchain support) by specifying them in square brackets:

```
pip install "generative-ai-hub-sdk[google, amazon]"
```

Configuration

There are different ways to configure the SAP AI Core access (listed in order of precedence):

- environment variables
- (profile) configuration file
- from VCAP_SERVICES environment variable, if it exists

These methods automatically initialize an authenticated client. For custom authentication, you can provide a `proxy_client` parameter when instantiating SDK classes to use your own `GenAIHubProxyClient` with direct credential configuration.

We recommend setting these values as environment variables or via config file. The default path for the configuration file is `~/.aicore/config.json`

Environment variables

- `AICORE_CLIENT_ID` : This represents the client ID.
- `AICORE_CLIENT_SECRET` : This stands for the client secret.
- `AICORE_AUTH_URL` : This is the URL used to retrieve a token using the client ID and secret.
- `AICORE_BASE_URL` : This is the URL of the service (with suffix /v2).
- `AICORE_RESOURCE_GROUP` : This represents the resource group that should be used.

For using X.509 credentials, you can set the file paths to certificate and key files, or certificate and key strings, as an alternative to client secret.

- `AICORE_CERT_FILE_PATH` : This is the path to the file which holds the X.509 certificate
- `AICORE_KEY_FILE_PATH` : This is the path to the file which holds the X.509 key
- `AICORE_CERT_STR` : This is the content of the X.509 certificate as a string
- `AICORE_KEY_STR` : This is the content of the X.509 key as a string

Configuration files

By default, the configuration file is located at `~/.aicore/config.json` . You can change the directory where the config file is located by setting the `AICORE_HOME` environment variable.

Note: tilde (~) is not supported, so use the full path to the directory.

A profile is a json file residing in a config directory. With profile names one can switch easily between profiles e.g., for different (sub)accounts. The profile name can be passed also as a keyword. If no profile is specified, the default profile is used. Specify the profile via envionment variable `AICORE_PROFILE` . The associated configuration file then needs to have file name `config_{profile}.json`

The command `ai-core-sdk configure --help` can be used to generate a profile.

The following list explains which environment variables can be used to control which configuration file will be used:

1. **`AICORE_HOME`** : This variable represents a directory path. Within this directory, various configuration files can be stored and the SDK will automatically load them from there based on the "AICORE_PROFILE" environment variable.

2. **`AICORE_PROFILE`** : This variable allows users to switch between different configurations stored in the `AICORE_HOME` directory. It is important to note that `AICORE_PROFILE` does not represent the complete name of a configuration file. Instead, it refers to a profile name, which corresponds to a file named `config_{profile}.json` . If AICORE_PROFILE is empty `$AICORE_HOME/config.json` is used.

3. **`AICORE_CONFIG`** : This variable overrides both `AICORE_HOME` and `AICORE_PROFILE` . It specifies the direct absolute path to a configuration file that will be used.

The configuration file should be:

```
{
  "AICORE_AUTH_URL": "https://* * * .authentication.sap.hana.ondemand.com",
  "AICORE_CLIENT_ID": "* * * ",
  "AICORE_CLIENT_SECRET": "* * * ",
  "AICORE_RESOURCE_GROUP": "* * * ",
  "AICORE_BASE_URL": "https://api.ai.* * *.cfapps.sap.hana.ondemand.com/v2"
}
```

```
{
  "AICORE_AUTH_URL": "https://* * * .authentication.cert.sap.hana.ondemand.c
  "AICORE_CLIENT_ID": "* * * ",
  "AICORE_CERT_FILE_PATH": "* * */cert.pem",
  "AICORE_KEY_FILE_PATH": "* * */key.pem",
  "AICORE_RESOURCE_GROUP": "* * * ",
  "AICORE_BASE_URL": "https://api.ai.* * *.cfapps.sap.hana.ondemand.com/v2"
}
```

or

```
{
  "AICORE_AUTH_URL": "https://* * * .authentication.cert.sap.hana.ondemand.c
  "AICORE_CLIENT_ID": "* * * ",
  "AICORE_CERT_STR": "* * *",
  "AICORE_KEY_STR": "* * *",
  "AICORE_RESOURCE_GROUP": "* * * ",
  "AICORE_BASE_URL": "https://api.ai.* * *.cfapps.sap.hana.ondemand.com/v2"
}
```

Usage

Prerequisite

For direct model access, you need to create a deployment for each desired model according to according to the help documentation for model deployments.

For model access through the orchestration service, you need to create a deployment of the orchestration service according to the help documentation for orchestration service deployments

Examples

In section "*Examples*" there are code snippets for each Large Language and Embedding model as well as for the orchestration service usage.

The list of models and scenarios in the Generative AI Hub of SAP AI Core can be found here. Among these, the following models are currently supported in the generative AI Hub SDK:

LLM Models

Provider	Model Name	Streaming Support
Amazon	amazon--nova-lite	No
	amazon--nova-micro	No
	amazon--nova-pro	No
	amazon--titan-text-express	Yes
	amazon--titan-text-lite	Yes
Anthropic	anthropic--claude-3-haiku	Yes
	anthropic--claude-3-opus	Yes
	anthropic--claude-3-sonnet	Yes
	anthropic--claude-3.5-sonnet	Yes
Google	gemini-1.0-pro	Yes
	gemini-1.5-flash	Yes
	gemini-1.5-pro	Yes
Meta	meta--llama3-70b-instruct	No
	meta--llama3.1-70b-instruct	No
MistralAI	mistralai--mixtral-8x7b-instruct-v01	No
	mistralai--mistral-large-instruct	No
OpenAI	gpt-4	Yes
	gpt-4o	Yes
	gpt-4o-mini	Yes
	gpt-4-32k	Yes
	ibm--granite-13b-chat	No
	o1	No

Provider	Model Name	Streaming Support
	o3-mini	No

Text Embedding Models

Provider	Model Name
Amazon	amazon--titan-embed-text
OpenAI	text-embedding-3-small
	text-embedding-3-large
	text-embedding-ada-002

Notes on model usage

- ⚠️ **Anthropic & Amazon**:
  - The `converse_stream` method is currently not supported for these models, use the `invoke_model_with_response_stream` method instead.
  - Currently, for `amazon--nova-lite`, `amazon--nova-micro`, and `amazon--nova-pro`, the supported method is `converse`. `invoke` and `invoke_model_with_response_stream` are not supported.
- ℹ️ **MistralAI:**
  - This model only supports the following roles in the order implied: user/assistant/user/assistant/....
- ℹ️ **Meta:**
  - These are also based on the OpenAI SDK and usage for these models is similar to that of GPT models.
- **Models not added to SDK yet**:
  - You can also try using Generative AI Hub SDK for models that are already in Generative AI Hub, but not supported yet by the SDK. This can be done by additionally specifying the model initialization: see Using New Models in Gen AI Hub Before They Are Added to SDK. Please note, that it's not guaranteed that it will

work. Because there might be some new models, for which customization in the SDK is needed.

Package dependencies

Please note the following dependencies of generative-ai-hub-sdk:

- ai_core_sdk>=2.5.7
- pydantic==2.9.2
- openai>=1.56.0
- google-cloud-aiplatform==1.61.0 # google
- boto3==1.35.76 # amazon
- langchain~=0.3.0 # langchain
- langchain-community~=0.3.0 # langchain
- langchain-openai>=0.2.14 # langchain
- langchain-google-vertexai==2.0.1 # langchain
- langchain-aws==0.2.9 # langchain