Integrate Semantic Kernel as a Service

Last Modified on 12/31/2024 10:49 am EST

V10.5 and later

Overview

Semantic Kernel is an open-source Software Development Kit (SDK) that enables you to build agents that can call any existing code and automates processes. With Semantic Kernel, you can create a Kernel object that provides your code to the AI. The Kernel then builds an agent that calls your code whenever its prompted.

Starting from V10.6, the Semantic Kernel supports Retrieval-Augmented Generation (RAG) capabilities. You can now configure the Azure AI Search and Qdrant connection to the "memoryType" as part of the Semantic Kernel configuration.
Azure AI Search is a powerful cloud-based search-as-a-service solution provided by Microsoft.
Qdrant is a vector search engine that provides a convenient API for managing vector data. It is designed for high-performance, real-time vector similarity search. It facilitates real time search capabilities and supports extended filtering.

Together with Gen AI these RAG capabilities can implement powerful search capabilities across internal documents, databases, and intranets, and support automated report generation, content summarization, and AI-driven insights extraction.

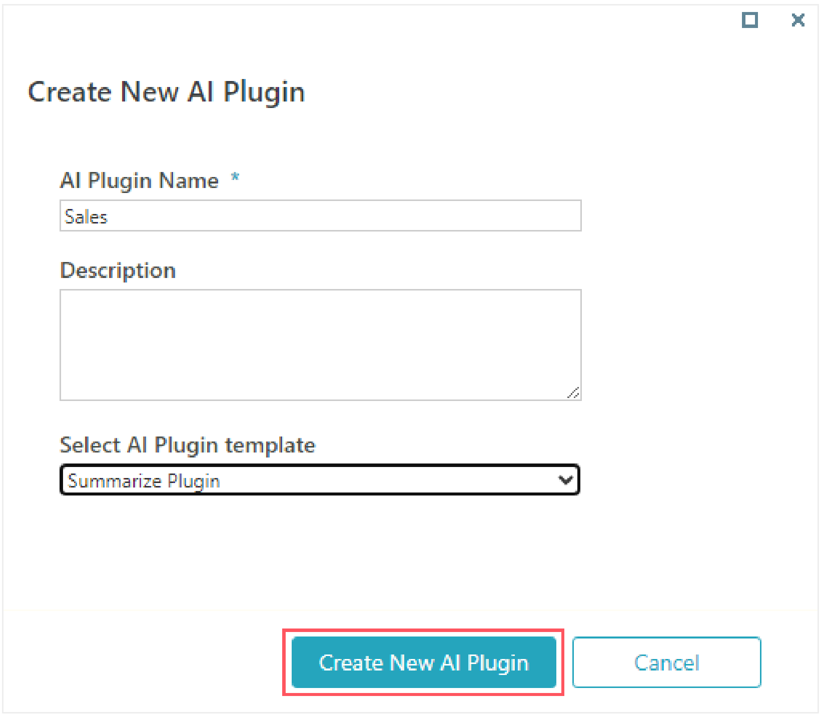
Configuration steps

	Steps	Location	Performed by
1	Create AI Plugin	Administration site	• Architect • Tech Lead • Developer
2	Create Semantic Kernel	Administration site	
3	Add SK (Semantic Kernel) function in workflow	Administration site>Workflow	

1. Create AI Plugin

Create AI Plugins to configure AI capabilities, prompt messages and other AI function parameters. A Plugin is a pre-defined set of functions that instructs the model on how to respond to user queries. A Plugin acts as a prompt template containing set of functions, and each function has two files `config.json` and `skprompt.txt`.

1. Go to **Administration** > **AI Services** > **AI Plugins**, and click **Add AI Plugin**.


Create New AI Plugin

AI Plugin Name *

Sales

Description

Select AI Plugin template

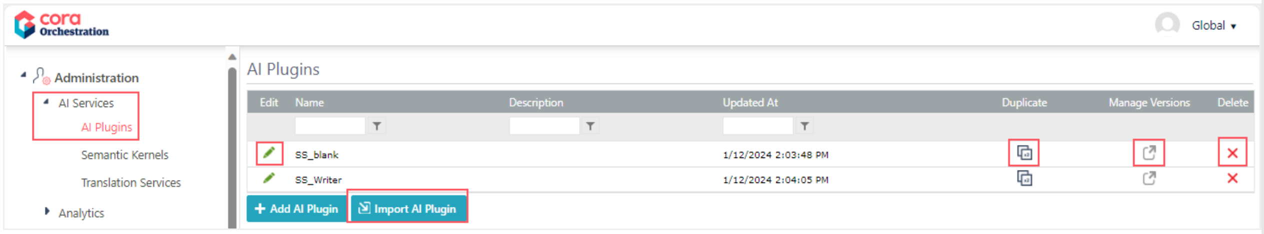
Summarize Plugin

2. Add Plugin name.
3. Select a Plugin template.
   Following are the three out-of-the-box Plugin templates provided:
   - Blank Plugin: The Plugin contains no pre-built functions so, you can add your own set of instructions.
   - Summarize Plugin: The Plugin contains functions that can help you summarize text.
   - Writer Plugin: The Plugin contains functions that can help you generate text, particularly for email response.
4. Click **Create New AI Plugin**.
   A Plugin editor window open up in a new tab, and based on your selection of the AI Plugin template a list of functions appear in the editor window.
5. Edit the config and prompt files in the functions per requirement.
6. Save the Plugin.

For configuration details, see this page.

Each AI Plugin can be used with different Kernels per requirement.

In the AI Plugins list, for a Plugin, click to edit and to delete the Plugin from the list. You can also create a duplicate and manage versions for the Plugin.



AI Plugins

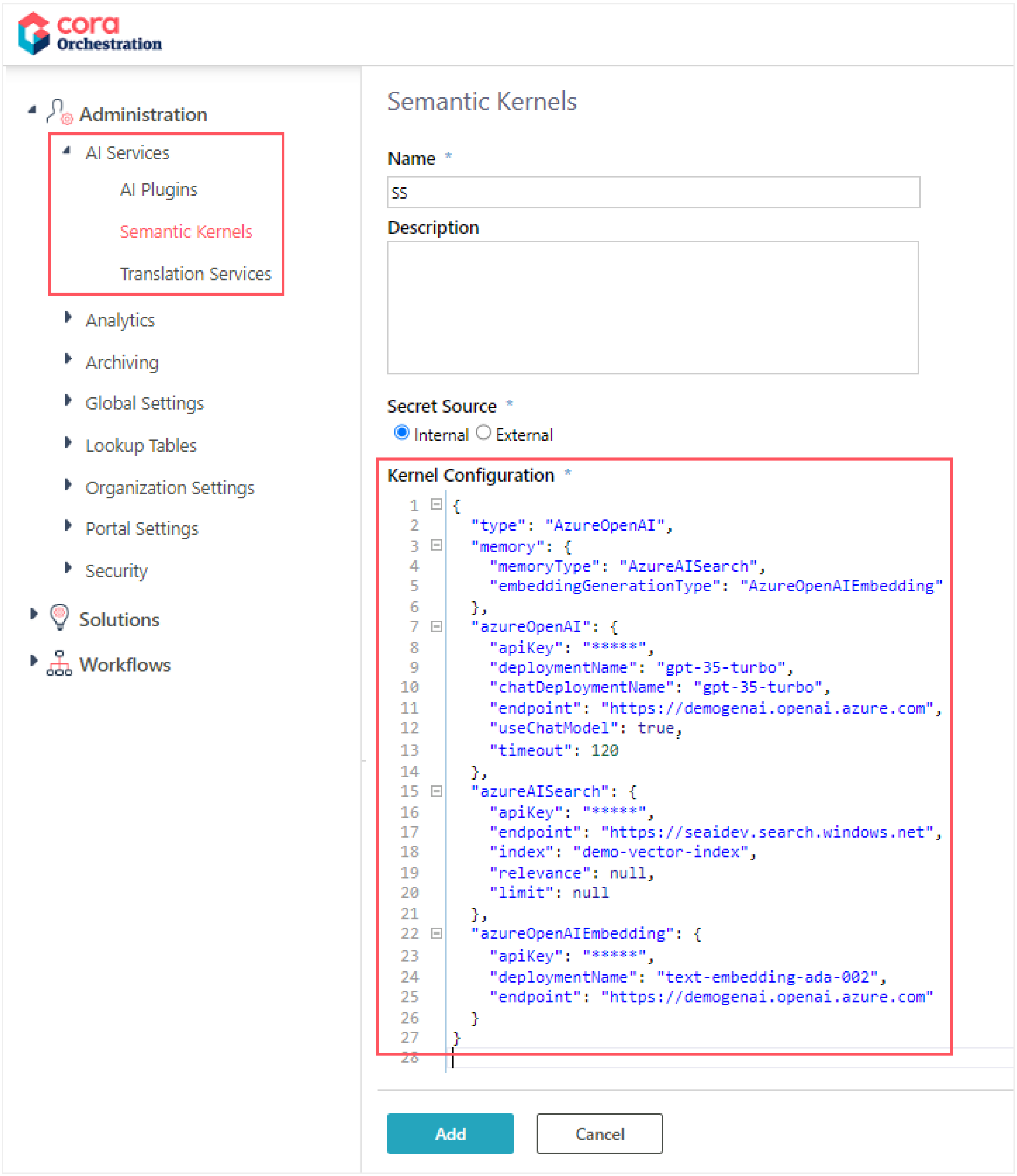
Edit	Name	Description	Updated At	Duplicate	Manage Versions	Delete
	SS_blank		1/12/2024 2:03:48 PM			
	SS_Writer		1/12/2024 2:04:05 PM			

Using the Import AI Plugin option, you can import any existing AI Plugin from a saved package to your system.

2. Create a Semantic Kernel

Create a Kernel object to connect to the AI engine.

1. Go to **Administration** > **AI Services** > **Semantic Kernels**, and click **Add New Record**.



Semantic Kernels

Name *
SS

Description

Secret Source *
[✓ Internal] External

Kernel Configuration *

```
{
  "type": "AzureOpenAI",
  "memory": {
    "memoryType": "AzureAISearch",
    "embeddingGenerationType": "AzureOpenAIEmbedding"
  },
  "azureOpenAI": {
    "apiKey": "*****",
    "deploymentName": "gpt-35-turbo",
    "chatDeploymentName": "gpt-35-turbo",
    "endpoint": "https://demogenai.openai.azure.com",
    "useChatModel": true,
    "timeout": 120
  },
  "azureAISearch": {
    "apiKey": "*****",
    "endpoint": "https://seaidev.search.windows.net",
    "index": "demo-vector-index",
    "relevance": null,
    "limit": null
  },
  "azureOpenAIEmbedding": {
    "apiKey": "*****",
    "deploymentName": "text-embedding-ada-002",
    "endpoint": "https://demogenai.openai.azure.com"
  }
}
```

2. Add a valid unique name for the Semantic Kernel.
3. Add a description.
4. Select the secret source for keys:
   - Internal: any internal source where keys are stored.
   - External: Azure or AWS.
5. Add the Kernel configuration.
6. Test the connection.
7. Click **Add**.

3. Add SK (Semantic Kernel) function in workflow

Once you have created an AI Plugin and a Kernel, you need to add a Semantic Kernel function to your workflow and configure the function.
For details, see this article.

Configure memoryType as part of Semantic Kernel configuration (V10.6)

Configuring the memoryType plugin in the Semantic Kernel configuration allows connection configuration to the vector database. For now, Qdrant and Azure AI Search are the two values you can select from for the memoryType.

Azure AI Search is the default memoryType.

The memoryType plugin supports a search function. The function contains the following parameters:

Parameter	Description
Input	The search value. (*Mandatory*)
Relevance	The relevance score from 0.0 to 1.0, where 1.0 means perfect match. (*Optional*)
Limit	The maximum number of relevant memories to search. (*Optional*)
Index	The memories collection to search. (*Mandatory*)

Syntax examples

- {{memoryPlugin.Search $input}}
- {{memoryPlugin.Search $input $limit=}}
- {{memoryPlugin.Search 'budget by year' index='finances'}} What is my budget for 2024?

Below are the sample configuration snippets for both.

Azure AI Search configuration

```
{
 "type": "AzureOpenAI",
 "memory": {
  "memoryType": "AzureAISearch",
  "embeddingGenerationType": "AzureOpenAIEmbedding"
 },

 "azureOpenAI": {
  "apiKey": "*****",
  "serviceId": null,
  "deploymentName": "Mandatory",
  "endpoint": "Mandatory",
  "useChatModel": true,
  "timeout": 120
 },

 "azureAISearch": {
  "apiKey": "*****",
  "endpoint": "Mandatory",
  "index": "Mandatory",
  "chatDeploymentName": "gpt-35-turbo",
  "relevance": null,
  "useChatModel": true,
  "limit": null
 },

 "azureOpenAIEmbedding": {
  "apiKey": "*****",
  "deploymentName": "Mandatory",
  "endpoint": "Mandatory"
 }
}
```

Example from V10.7.1 onwards

```
{
  "type": "AzureOpenAI",
  "memory": {
    "memoryType": "AzureAISearch",
    "embeddingGenerationType": "AzureOpenAIEmbedding"
  },

  "azureOpenAI": {
    "apiKey": "*****",
    "deploymentName": "gpt-35-turbo",
    "endpoint": "https://*****.openai.azure.com",
    "timeout": 120
  },

  "azureAISearch": {
    "apiKey": "*****",
    "endpoint": "https://*****.search.windows.net",
    "index": "demo-vector-index",
    "relevance": null,
    "limit": 2
  },

  "azureOpenAIEmbedding": {
    "apiKey": "*****",
    "deploymentName": "text-embedding-ada-002",
    "endpoint": "https://*****.openai.azure.com"
  }
}
```

Qdrant configuration

```
  "qdrant": {
    "endpoint": "Mandatory",
    "size": 1536,
    "index": "Mandatory",
    "relevance": null,
    "limit": null
  },
```