Integrations

In addition to the ILIAD chat and embed endpoints, we also support integration with third-party SDKs and libraries such as LangChain, OpenAI, and Anthropic.

OpenAI Integrations

Model names

To choose an OpenAI model, please supply one of the following values:

- gpt-4o
- gpt-4o-global
- gpt-4o-mini
- gpt-4o-mini-global
- gpt-4-turbo
- text-embedding-ada-002
- text-embedding-3-large
- text-embedding-3-small
- o3-global
- o4-mini-global
- gpt-4.1-global
- gpt-4.1-nano-global
- gpt-4.1-mini-global
- gpt-4.1
- gpt-4.1-nano
- gpt-4.1-mini
- gpt-4o-mini-tts
- gpt-5-global
- gpt-5-nano-global
- gpt-5-mini-global
- gpt-5-chat-global

LangChain OpenAI

You'll need to install two Python packages: langchain and langchain-openai.

```
pip install langchain langchain-openai
```

Note

This example only works with langchain version 1.0.0 and greater.

Store your Iliad API key as an environment variable (see our guide).

Choose your model by supplying a model name to the `azure_deployment` parameter of the `AzureChatOpenAI` class.

LangChain OpenAI chat

```
from langchain.schema import HumanMessage
from langchain_openai import AzureChatOpenAI, AzureOpenAIEmbeddings

model = AzureChatOpenAI(
    api_key=ILIAD_KEY,
    azure_endpoint="https://api-epic.ir-gateway.abbvienet.com/iliad",
    openai_api_version="2023-07-01-preview",
    azure_deployment="gpt-4o",
)

message = HumanMessage(content="Hello GPT!")

completion = model.invoke([message])

print(completion.content)
```

LangChain OpenAI embeddings

```
from langchain_openai import AzureOpenAIEmbeddings

embedding_model = AzureOpenAIEmbeddings(
    api_key=ILIAD_KEY,
    azure_endpoint="https://api-epic.ir-gateway.abbvienet.com/iliad",
    openai_api_version="2023-07-01-preview",
)

embeddings = embedding_model.embed_documents(["this is a test"])
```

OpenAI Python library

You’ll need to install the openai python library:

```
pip install openai
```

Note

This example only works with openai python library version 1.0.0 and greater.

Store your Iliad API key as an environment variable (see our guide).

OpenAI Chat

```
from openai import AzureOpenAI

client = AzureOpenAI(
    azure_endpoint="https://api-epic.ir-gateway.abbvienet.com/iliad",
    api_key=ILIAD_KEY,
    api_version="2023-07-01-preview",
)

response = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "Does Azure OpenAI support customer managed keys?"},
        {"role": "assistant", "content": "Yes, customer managed keys are supported by Azure
OpenAI."},
        {"role": "user", "content": "Do other Azure AI services support this too?"}
    ]
)

print(response.choices[0].message.content)
```

OpenAI Speech-to-Text

```
from openai import AzureOpenAI

client = AzureOpenAI(
    base_url=f"{BASE_URL}/openai/deployments/whisper-1/audio/transcriptions?api-version=2024-02-
15-preview",
    api_key=ILIAD_KEY,
    api_version="2024-02-15-preview",
)
with open("path/to/audio_file.mp3", "rb") as audio_file:
    transcription = client.audio.transcriptions.create(
        model="whisper-1",
        file=audio_file,
        response_format="text"  # Options: "json", "text", "srt", "vtt", or "verbose_json"
    )

print(transcription)
```

Anthropic integrations

Models

To choose an Anthropic model, please supply one of the following values from the *model name* column:

model	model name
Claude 3 Sonnet	`claude-3-sonnet-20240229`
Claude 3 Haiku	`claude-3-haiku-20240307`
Claude 3 Opus	`claude-3-opus-20240229`
Claude 3.5 Haiku	`claude-3-5-haiku-20241022`
Claude 3.5 Sonnet (v1)	`claude-3-5-sonnet-20240620`
Claude 3.5 Sonnet (v2)	`claude-3-5-sonnet-20241022`
Claude 3.7 Sonnet	`claude-3-7-sonnet-20250219`
Claude 4 Sonnet	`claude-sonnet-4-20250514`
Claude 4 Opus	`claude-opus-4-20250514`

LangChain Anthropic chat

You'll need to install the anthropic python library:

```
pip install langchain-anthropic
```

Store your Iliad API key as an environment variable (see our guide).

```
from langchain_anthropic import ChatAnthropic
from langchain.schema import HumanMessage

model = ChatAnthropic(
    anthropic_api_url="https://api-epic.ir-gateway.abbvienet.com/iliad/anthropic",
    api_key=ILIAD_KEY,
    model_name="claude-3-haiku-20240307"
)

prompt = HumanMessage(content="Hello Claude!")

completion = model.invoke([prompt])

print(completion.content)
```

Langchain's AWS integration is required to stream, you can install it via

```
pip install langchain-aws
```

```
from langchain_aws import ChatBedrock
from pydantic import SecretStr

client = ChatBedrock(
    endpoint_url="https://api-epic.ir-gateway.abbvienet.com/iliad/anthropic",
    model="claude-3-5-haiku-20241022",
    provider="anthropic",
    aws_access_key_id=SecretStr("dummy"),
    aws_secret_access_key=SecretStr("dummy"),
    region="us-west-1"
)
messages = [
    (
        "system",
        "You are a helpful assistant that translates English to French. Translate the user
sentence.",
    ),
    ("human", "I love programming."),
]
ai_msg = client.stream(messages)
for m in ai_msg:
    print(m)
```

Anthropic Python library

You'll need to install the Anthropic python library:

```
pip install anthropic
```

Store your Iliad API key as an environment variable (see our guide).

```
import anthropic

ILIAD_URL = "https://api-epic.ir-gateway.abbvienet.com/iliad/anthropic"

client = anthropic.Anthropic(base_url=ILIAD_URL, api_key=ILIAD_KEY)

message = client.messages.create(
    model="claude-3-haiku-20240307",
    max_tokens=100,
    messages=[{"role": "user", "content": "Hello, Claude!"}]
)

print(message.content[0].text)
```

To stream:

```
import anthropic

ILIAD_URL = "https://api-epic.ir-gateway.abbvienet.com/iliad/anthropic"

client = anthropic.Anthropic(base_url=ILIAD_URL, api_key=ILIAD_KEY)

message = client.messages.create(
    model="claude-3-haiku-20240307",
    max_tokens=100,
    messages=[{"role": "user", "content": "Hello, Claude!"}],
    stream=True
)

for event in message._iter_events():
    print(event.json())
```

Users can now stream responses without needed to call *._iter_events()*. To do this, use the *AnthropicBedrock* client:

```python
import anthropic

client = anthropic.AnthropicBedrock(
    base_url=ILIAD_URL + "/anthropic",
)

message = client.messages.create(
    model="claude-3-haiku-20240307",
    max_tokens=100,
    messages=[{"role": "user", "content": "Hello, Claude!"}],
    stream=True
)

for event in message:
    print(event)
```