Use the OpenAI Python SDK with Radium

Radium exposes an OpenAI-compatible Chat Completions API. If your application already uses the OpenAI Python SDK, you only need to change the API key, base URL, and model name.

This guide creates a small program that sends one request and prints the response.

Before you start

You need:

- Python 3.10 or newer
- A Radium API key
- A terminal or PowerShell window

1. Create a project and install the SDK

macOS or Linux:

```
mkdir radium-openai-python
cd radium-openai-python
python3 -m venv .venv
source .venv/bin/activate
python -m pip install openai
```

Windows PowerShell:

```
mkdir radium-openai-python
cd radium-openai-python
py -m venv .venv
.venv\Scripts\Activate.ps1
python -m pip install openai
```

2. Set your Radium credentials

macOS or Linux:

```
export RADIUM_API_KEY="your-radium-api-key"
export RADIUM_MODEL="hal-1.0"
```

Windows PowerShell:

```
$env:RADIUM_API_KEY = "your-radium-api-key"
$env:RADIUM_MODEL = "hal-1.0"
```

Keep the key on the server. Do not paste it into `main.py`, commit it, or expose it in browser code.

3. Create `main.py`

```
import os

from openai import OpenAI



api_key = os.getenv("RADIUM_API_KEY")
if not api_key:
    raise SystemExit("Set RADIUM_API_KEY before running this program.")

model = os.getenv("RADIUM_MODEL", "hal-1.0")
client = OpenAI(
    api_key=api_key,
    base_url="https://api.radium.cloud/v1",
    timeout=90,
    max_retries=0,
)

response = client.chat.completions.create(
    model=model,
    messages=[
        {"role": "system", "content": "Follow the user's instruction exactly."},
        {"role": "user", "content": "Reply with exactly: Radium connected."},
    ],
```

```
    max_tokens=512,
    temperature=0,
)

content = response.choices[0].message.content
if not content or not content.strip():
    raise RuntimeError("Radium returned no visible text.")
print(f"RADIUM_RESPONSE: {content.strip()}")
```

The base URL must include `/v1`. Do not append `/chat/completions`; the SDK adds that path.

4. Run it

```
python main.py
```

Expected output:

```
RADIUM_RESPONSE: Radium connected.
```

Choose a model

Set `RADIUM_MODEL` to any validated model before running the program:

```
export RADIUM_MODEL="hal-1.0"
# or: clarke-1.0
# or: tycho-1.0
python main.py
```

Migrate an existing OpenAI client

Before:

```
client = OpenAI(api_key=os.environ["OPENAI_API_KEY"])
```

After:

```
client = OpenAI(
    api_key=os.environ["RADIUM_API_KEY"],
```

```
    base_url="https://api.radium.cloud/v1",
)
```

Keep your existing messages and response parsing. Replace the old model ID with `hal-1.0`, `clarke-1.0`, or `tycho-1.0`.

Troubleshooting

- `401` or `403`: verify that `RADIUM_API_KEY` is set in the same terminal running Python.
- `404`: verify the base URL and use one of the model IDs listed above.
- Empty output: keep `max_tokens` at 512 or higher so model reasoning does not consume the visible response allowance.
- `ModuleNotFoundError`: activate `.venv` and rerun the installation command.

Validation

Version note: these instructions were verified with Python 3.12.7 and `openai==2.54.0` on August 20, 2026. You do not need that exact package version. Text generation and a complete function-calling loop passed with `hal-1.0`, `clarke-1.0`, and `tycho-1.0`.

Reference: OpenAI Python library.

Next

- API quickstart, for calling Radium directly
- Tool calling, for the `tool_use` and `tool_result` contract

Created 2026-08-23 21:37:13 UTC by Admin
Updated 2026-08-23 21:53:58 UTC by Admin