The recommended way to integrate a custom LLM with Portalis AI avatars is using the OpenAI API with Server Side Events (SSE). This allows us to integrate your LLM quickly, and provide the fastest possible speech using an SSE streamed response. Custom integrations will take more time and manpower than an interface that follows the OpenAI standard. Non-streamed responses will cause significant delays in between sections of avatar speech.

Below is the format of the request that Portalis will send:

```
Unset
{
    "model": "gpt-4o",
    "messages": [
      {
        "role": "developer",
        "content": "You are a helpful assistant."
      },
      {
        "role": "user",
        "content": "Hello!"
      }
    ],
    "stream": true
  }
```

And the expected response:

```
Unset
{"id":"chatcmpl-123","object":"chat.completion.chunk","created":1694268190,"mod
el":"gpt-4o-mini", "system_fingerprint": "fp_44709d6fcb",
"choices":[{"index":0,"delta":{"role":"assistant","content":""},"logprobs":null
,"finish_reason":null}]}

{"id":"chatcmpl-123","object":"chat.completion.chunk","created":1694268190,"mod
el":"gpt-4o-mini", "system_fingerprint": "fp_44709d6fcb",
"choices":[{"index":0,"delta":{"content":"Hello"},"logprobs":null,"finish_reaso
n":null}]}

{"id":"chatcmpl-123","object":"chat.completion.chunk","created":1694268190,"mod
el":"gpt-4o-mini", "system_fingerprint": "fp_44709d6fcb",
"choices":[{"index":0,"delta":{},"logprobs":null,"finish_reason":"stop"}]}
```

In the previous example, the sum total of the conversation is as follows:

System Prompt: You are a helpful assistant
User: Hello!
Assistant: Hello

Example python script for testing an OpenAI API compliant endpoint. Change the URL and Key as it applies.

Python

```python
import requests
import json

# Replace with your OpenAI API key
API_KEY = "your_openai_api_key"

# OpenAI API endpoint for ChatGPT
API_URL = "https://api.openai.com/v1/chat/completions"

def chat_with_gpt(prompt):
    headers = {
        "Authorization": f"Bearer {API_KEY}",
        "Content-Type": "application/json"
    }

    data = {
        "model": "gpt-4",  # Use "gpt-3.5-turbo" if you prefer
        "messages": [
            {"role": "system", "content": "You are a helpful assistant."},
            {"role": "user", "content": prompt}
        ],
        "stream": True
    }

    response = requests.post(API_URL, headers=headers, data=json.dumps(data),
stream=True)

    if response.status_code == 200:
        for line in response.iter_lines():
            if line:
                decoded_line = json.loads(line.decode("utf-8").replace("data:
", ""))
                if "choices" in decoded_line:
                    print(decoded_line["choices"][0]["delta"].get("content",
""), end="", flush=True)
    else:
        print(f"Error: {response.status_code}, {response.text}")
```

```
if __name__ == "__main__":
    user_input = input("Enter your message: ")
    chat_with_gpt(user_input)
```