

ai-**PULSE** by  Scaleway

# Agentic AI powered by Generative APIs: Building high-precision function calling on open-weight LLMs

# Hello, I'm Guillaume 🖐️

@ AI Inference Team



Managed inference



Generative APIs



# LLMs: going beyond simple chat



# What we will cover in this session

## 01 From simple chat to agentic pattern

How structured outputs and function calling can be leveraged for integrating LLMs into an Enterprise ecosystem

## 02 Generative APIs

Use Scaleway open-weights based inference service as a drop-in replacement for all apps leveraging OpenAI APIs

## 03 Hands-on code and demo

Let's dive in with some concrete examples

# Notes on agentic patterns

The agentic concept in LLM-based applications refers to developing AI systems that **can act autonomously**, make decisions and **perform tasks with minimal human intervention**.

These “agents” leverage large language models (LLMs) to understand context, generate responses and take actions based on given objectives.

👉 This session will principally focus on the **Function Calling** agentic pattern



# The path to agentic LLM interaction

## Leverage information stored in LLMs weights

QA general knowledge,  
Summarization task

## Leverage external knowledge provided as context to LLMs

Knowledge base discovery,  
RAG, automated meeting notes

## Provide access to existing tools and systems to LLMs

Programmatic interaction,  
Agentic patterns



Domain specific  
knowledge



IT system  
integration

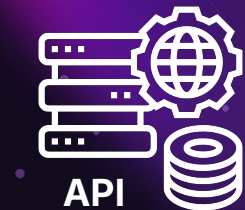


**“Generating structured data from unstructured inputs is one of the core use cases for AI in today’s applications”**



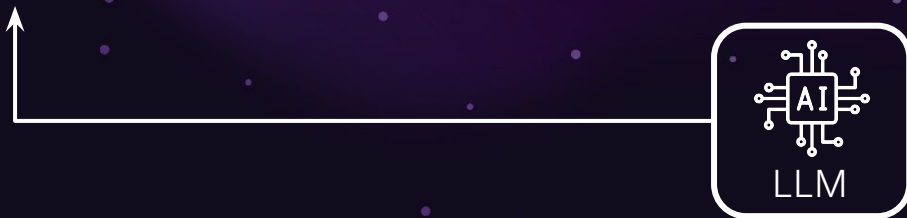
08/2024

# Structured outputs for reliable applications



API  
Interface

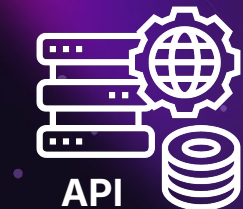
Applications and systems have interfaces with specific inputs and arguments



LLMs outputs **must be structured** and predictable to be used in downstream applications, typically using **JSON**

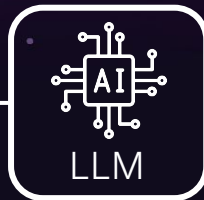


# Structured outputs for reliable applications



API  
Interface

Applications and systems have interfaces with specific inputs and arguments



LLMs outputs **must be structured** and predictable to be used in downstream applications, typically using **JSON**

But asking for JSON in the past has often been unreliable ...

## “What are the 2 best NBA players in history ? Format your response as JSON.”

Here are the two best NBA players in history, based on their impact, statistics, and accolades. Please note that the “best” players can be subjective and depend on personal preferences. This list is based on widespread consensus among basketball experts and fans.

```
```json
[
  {
    "name": "Michael Jordan",
    "position": "Shooting Guard",
    "team": "Chicago Bulls",
    "achievements": ["6x NBA Champion", "5x NBA MVP", "14x NBA All-Star",
      "10x NBA Scoring Leader", "2x Olympic Gold Medalist",
      "NBA All-Time Leading Scorer (32,292 points)"]
  },
  {
    "name": "LeBron James",
    "position": "Small Forward",
    "team": "Los Angeles Lakers",
    "achievements": [
      "4x NBA Champion", "4x NBA MVP", "17x NBA All-Star",
      "NBA All-Time Leading Scorer (38,397 points)",
      "4x NBA Finals MVP"
    ]
  }
]
```
```

“Just the JSON please!”

```
{
  "player1": {
    "name": "Michael Jordan",
    "position": "Shooting Guard",
    "team": "Chicago Bulls",
    "achievements": [
      "6x NBA Champion",
      "5x NBA MVP",
      "10x NBA Scoring Leader",
      "2x Olympic Gold Medalist"
    ]
  },
  "player2": {
    "name": "LeBron James",
    "position": "Small Forward",
    "team": "Los Angeles Lakers",
    "achievements": [
      "4x NBA Champion",
      "4x NBA MVP",
      "17x NBA All-Star",
      "2x Olympic Gold Medalist"
    ]
  }
}
```

“What are the 2 best NBA players in history ? Format your response as JSON.”

Here are the two best NBA players in history, based on their impact, statistics, and accolades. Please note that the “best” players can be subjective and depend on personal preferences. This list is based on widespread consensus among basketball experts and fans.

```
```json
[
  {
    "name": "Michael Jordan",
    "position": "Shooting Guard",
    "team": "Chicago Bulls",
    "achievements": ["6x NBA Champion", "5x NBA MVP", "14x NBA All-Star",
      "10x NBA Scoring Leader", "2x Olympic Gold Medalist",
      "NBA All-Time Leading Scorer (32,292 points)"]
  },
  {
    "name": "LeBron James",
    "position": "Small Forward",
    "team": "Los Angeles Lakers",
    "achievements": [
      "4x NBA Champion", "4x NBA MVP", "17x NBA All-Star",
      "NBA All-Time Leading Scorer (38,397 points)",
      "4x NBA Finals MVP"
    ]
  }
]
```
```



json output



inconsistent schema

“Just the JSON please!”

```
{
  "player1": {
    "name": "Michael Jordan",
    "position": "Shooting Guard",
    "team": "Chicago Bulls",
    "achievements": [
      "6x NBA Champion",
      "5x NBA MVP",
      "10x NBA Scoring Leader",
      "2x Olympic Gold Medalist"
    ]
  },
  "player2": {
    "name": "LeBron James",
    "position": "Small Forward",
    "team": "Los Angeles Lakers",
    "achievements": [
      "4x NBA Champion",
      "4x NBA MVP",
      "17x NBA All-Star",
      "2x Olympic Gold Medalist"
    ]
  }
}
```

# Towards reliable structured outputs

## Constrain which tokens can be produced next:

The LLM still generates probabilities for likely next tokens, but any tokens that would not match the desired schema/grammar is masked



Introduced in the  
API (08/2024)

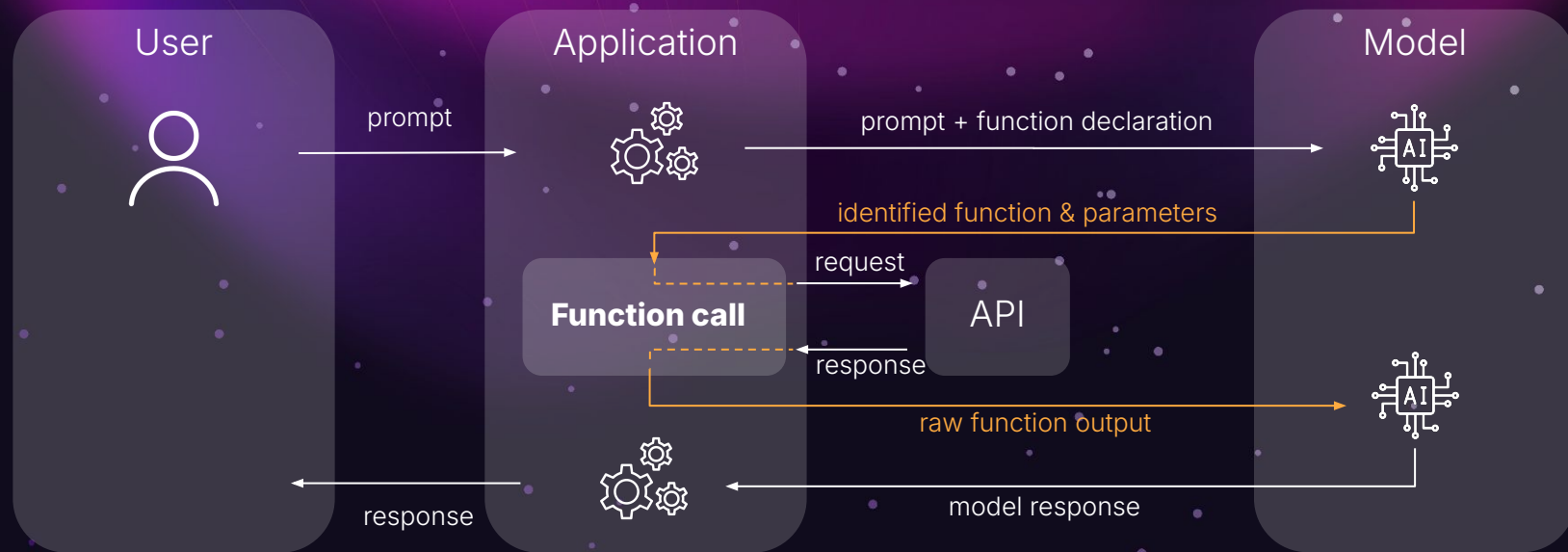
```
"response_format": {  
  "type": "json_object"  
}
```

ensures that the output  
will be valid JSON

```
"response_format": {  
  "type": "json_schema",  
  "json_schema": { ... }  
}
```

ensures that the output will exactly  
match a specified JSON schema

# Function calling relies on structured outputs



LLM can't directly query or modify external data.  
The basic idea is to describe tools (functions & signatures) so that LLMs can generate proper **structured output** to invoke those external APIs.



# Generative APIs

OpenAI APIs drop-in replacement for your open-weight-based inference needs

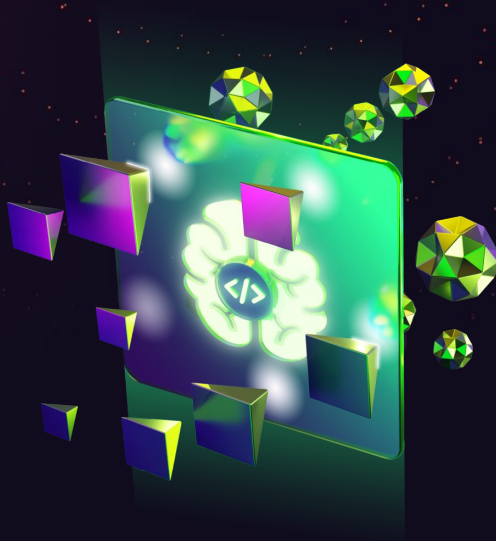
Now in free public beta

Open weights

Sovereign cloud

OpenAI APIs compatible

Agentic pattern ready



**Instruct**

Llama 3.1 8B

Llama 3.1 70B



**Vision**

Nemo Instruct 2407

Pixtral 12B



**Embeddings**

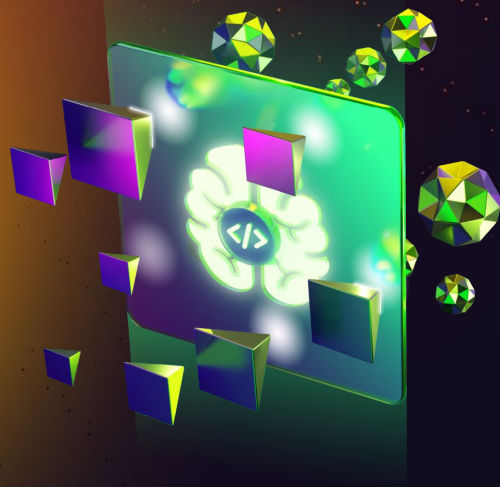
BGE-Multilingual-Gemma2

Sentence T5 XL



# Generative APIs

OpenAI APIs drop-in replacement for your open-weight-based inference needs



```
from openai import OpenAI

client = OpenAI(
    base_url="https://api.scaleway.ai/v1",
    api_key=SCW_API_KEY,
)

response = client.chat.completions.create(
    model = "llama-3.1-8b-instruct",
    messages = [
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "Tell me a joke about a frog."}
    ],
)
```

# Hands on & demo