Documentation

Flex Processing

Flex Processing

Flex Processing is a service tier optimized for high-throughput workloads that prioritizes fast inference and can handle occasional request failures. This tier offers significantly higher rate limits while maintaining the same pricing as on-demand processing during beta.

Availability

Flex processing is available for all models to paid customers only with 10x higher rate limits compared to on-demand processing. While in beta, pricing will remain the same as our on-demand tier.

Service Tiers

- **On-demand (`"service_tier":"on_demand"`):** The on-demand tier is the default tier and the one you are used to. We have kept rate limits low in order to ensure fairness and a consistent experience.
- **Flex (`"service_tier":"flex"`):** The flex tier offers on-demand processing when capacity is available, with rapid timeouts if resources are constrained. This tier is perfect for workloads that prioritize fast inference and can gracefully handle occasional request failures. It provides an optimal balance between performance and reliability for workloads that don't require guaranteed processing.
- **Auto (`"service_tier":"auto"`):** The auto tier uses on-demand rate limits, then falls back to flex tier if those limits are exceeded.

Using Service Tiers

Service Tier Parameter

The `service_tier` parameter is an additional, optional parameter that you can include in your chat completion request to specify the service tier you'd like to use. The possible values are:

OPTION	DESCRIPTION
`flex`	Only uses flex tier limits
`on_demand` (default)	Only uses on_demand rate limits

OPTION	DESCRIPTION
`auto`	First uses on_demand rate limits, then falls back to flex tier if exceeded

Example Usage

curl JavaScript **Python** JSON

```python
import os
import requests

GROQ_API_KEY = os.environ.get("GROQ_API_KEY")

def main():
    try:
        response = requests.post(
            "https://api.groq.com/openai/v1/chat/completions",
            headers={
                "Content-Type": "application/json",
                "Authorization": f"Bearer {GROQ_API_KEY}"
            },
            json={
                "service_tier": "flex",
                "model": "llama-3.3-70b-versatile",
                "messages": [{
                    "role": "user",
                    "content": "whats 2 + 2"
                }]
            }
        )
        print(response.json())
    except Exception as e:
        print(f"Error: {str(e)}")

if __name__ == "__main__":
    main()
```