大模型服务平台 TokenHub

实践教程

腾讯云

【版权声明】

【商标声明】

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

实践教程
TokenHub 迁移指南

最近更新时间：2026-05-11 17:48:31

为提升用户使用大模型的体验，原混元大模型、知识引擎原子能力—DeepSeek API（**以下简称：原平台**）大模型售卖入口将迁移到 TokenHub。原平台将不再新增模型能力，并停止新购模型服务，详情以平台下线公告通知为准，建议您将已购买的模型服务迁移到 TokenHub 使用。

迁移收益

TokenHub 支持 DeepSeek V4 Pro/Flash、Hy3-preview、GLM 5.1、Kimi-K2.6、MiniMax-M2.7 等更多、更新的行业热门模型，和原平台相比可选范围更丰富，详情请参见 模型列表。
TokenHub 为首次使用本产品的用户提供了多款模型的免费使用额度，详情请参见 新人免费体验包。

涉及范围

1. 原平台已开启后付费的用户

对于开启了混元、DeepSeek 后付费的用户，可直接迁移到 TokenHub，操作步骤请参见 迁移方式。

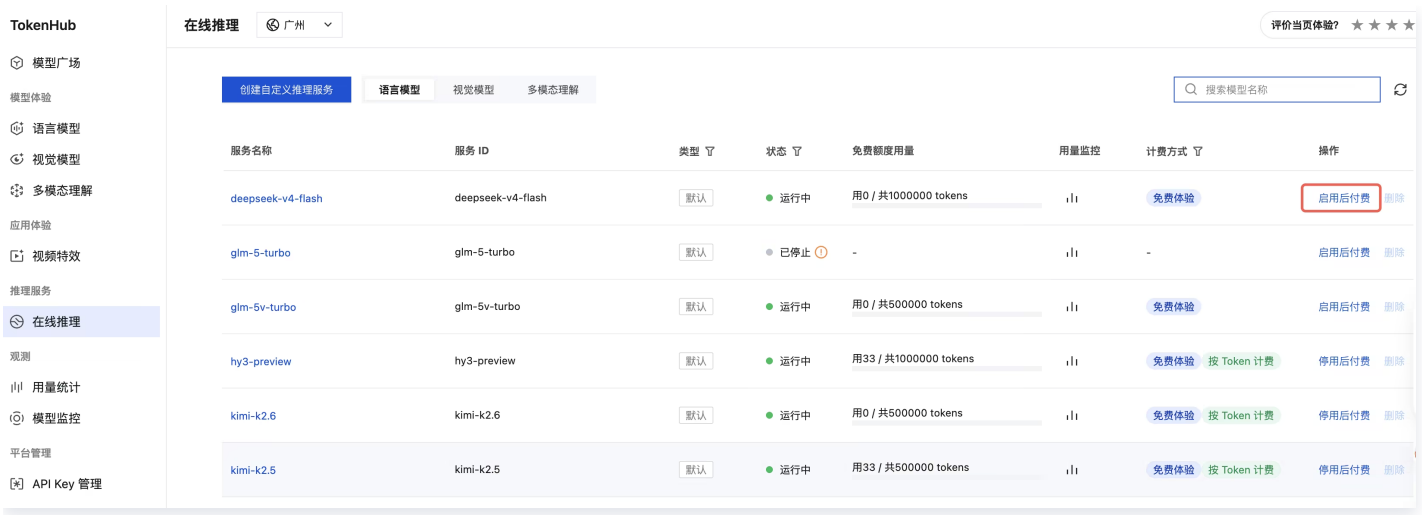
2. 原平台已开启预付费的用户

TokenHub 将不再支持预付费模式的 Token 资源包购买，您可以选择在原有资源包用完后切换至 TokenHub，或联系平台协助完成按未使用资源比例的退费后切换至 TokenHub。

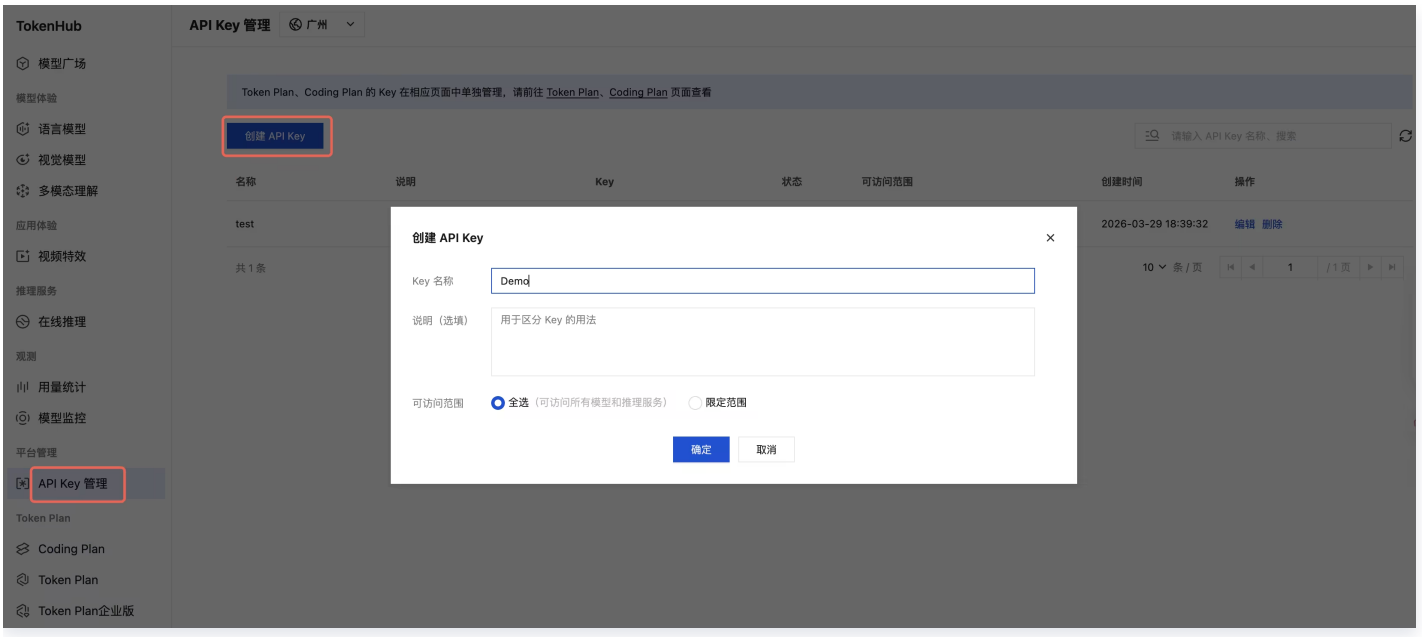
> ⚠ **注意：**
> - 退费计算方式：退费金额=资源包购买金额*（1-资源包使用比例），退款通常需要数天至1 周原路退回，详细信息需以退款申请审核通知为准。
> - 对于享受购买折扣的用户，平台将协助完成对应折扣转移到 TokenHub。
> - hunyuan-t1-latest、hunyuan-a13b、hunyuan-turbos-latest、hunyuan-lite、hunyuan-translation、hunyuan-translation-lite、hunyuan-large-role-latest，TokenHub 将不再支持，建议您切换到 TokenHub 时改为使用更新的模型。
> - 原平台联网搜索能力将随平台一同下线。TokenHub 将上线全新的联网搜索能力，与原平台为不同产品，接口协议、使用方式和定价均有变化，详情请参见 附录。

迁移方式

第一步：登录 TokenHub 控制台，前往 在线推理 页面，选择您需要的模型，开启**免费体验**以及**启用后付费**，开启后将优先消耗免费体验额度，免费体验额度使用完之后按 Token 用量后付费，模型后付费价格请参见 模型价格。



第二步：在 TokenHub API Key 管理 页面中创建新的 API Key。操作详情请参见 API Key 管理 。



第三步：参考 模型详情页 中的调用示例（以 DeepSeek V4 Flash 为例），更换调用 DeepSeek API 的 URL 和 API Key。

- 若通过 OpenAI SDK 访问，则 `base_url` 更换为：`https://tokenhub.tencentmaas.com/v1`
- 若通过 URL 直接访问，则 URL 为：`https://tokenhub.tencentmaas.com/v1/chat/completions`

更多 API 调用示例请参见 API 使用说明 。

联系我们

如果您在迁移过程中遇到任何问题或需要帮助，您可以咨询 在线客服 或 提交工单 来与我们联系。

附录

联网搜索功能新旧平台对比

维度	原平台（将下线）	TokenHub（全新）
产品定位	原有附属功能，随平台下线	全新独立产品能力
API 协议	Chat Completions	Responses API
接口参数	enable_search: true	全新参数体系（上线时发布）
定价	• DeepSeek 联网搜索服务：后付费 8元/千次。 • 腾讯混元大模型：未单独收费。	全新定价体系（上线时公布）
关键时间	跟随原平台下线	Hy3-preview 模型预计 5 月底支持联网搜索，DeepSeek 等热门模型预计 6 月底上线联网搜索。

如您在原平台使用了联网搜索，迁移至 TokenHub 后需按照新的接口和定价重新接入。原平台联网搜索在停服前仍可正常使用。

提升 Cache 命中率指南

最近更新时间：2026-05-11 22:06:01

本文档介绍如何通过合理的请求设计，提升 TokenHub 平台的 Prompt Cache 命中率，从而降低首 Token 响应时间（TTFT）和推理成本。

一、使用 prompt_cache_key

1.1 什么是 prompt_cache_key

`prompt_cache_key` 是请求级别的缓存标识字段，用于告诉缓存系统哪些请求的前缀是相同的，可以复用 KV Cache。
其核心原则是：**赋值为整体上下文总 ID（conversation_id），而非单一会话 ID（session_id）。**

1.2 使用方式

在请求体中添加 `prompt_cache_key` 字段：

```
{
  "model": "your-model",
  "prompt_cache_key": "conv-6900xxxx",
  "messages": [
    {"role": "system", "content": "你是一个助手..."},
    {"role": "user", "content": "你好"}
  ]
}
```

1.3 最佳实践

- 同一对话上下文的所有请求使用**相同的** `prompt_cache_key` 。
- 不同对话使用**不同的** `prompt_cache_key` ，避免缓存污染。
- 值建议使用业务侧的 conversation_id 或等价的全局唯一标识。

二、使用 X-Session-ID

2.1 什么是 X-Session-ID

`X-Session-ID` 是通过 HTTP Header 传递的会话标识，用于将同一用户的连续请求路由到同一个推理实例，从而提高该实例上的 KV Cache 局部命中率。

2.2 使用方式

在请求 Header 中添加：

```
X-Session-ID: session-abc123
```

完整请求示例：

```
curl -X POST 'https://tokenhub.tencentmaas.com/v1/chat/completions' \
  -H 'Content-Type: application/json' \
  -H 'Authorization: Bearer your-api-key' \
  -H 'X-Session-ID: session-abc123' \
  -d '{
    "model": "your-model",
    "messages": [...]
  }'
```

2.3 最佳实践

- 同一用户的多轮对话，保持 `X-Session-ID` 不变。
- 不同用户或不同对话上下文，使用不同的 Session ID。
- 配合 `prompt_cache_key` 一起使用效果更佳。

三、缓存 TTL 机制

3.1 什么是缓存 TTL

TTL（Time To Live）是缓存的存活时间。超过 TTL 后，缓存的 KV 数据将被淘汰，后续请求需要重新计算。

3.2 TTL 对命中率的影响

在 TTL 有效期内，相同前缀的请求可直接复用已缓存的 KV 数据，TTL 过期后，即使前缀相同也需要重新计算，导致 TTFT 回升。**用户可以通过优化请求设计，避免缓存被意外提前失效**，从而在 TTL 有效期内获得最大命中收益。核心关注点：**保持请求前缀稳定，不要让前缀中的内容频繁变化。**

3.3 注意事项

避免 System Prompt 中写入时间相关内容

> ⚠ **注意：**
> 不要在 System Prompt 中写入当天日期、当前时间等动态内容。

原因：时间变化会导致 System Prompt 内容变更 → 前缀不匹配 → 缓存完全失效。例如，过了 0 点日期变更，瞬间所有缓存失效，TTFT 暴涨，用户体验为"特别卡"。

正确做法：

```
// ✕ 错误：system prompt 包含动态时间
{
  "messages": [
    {"role": "system", "content": "今天是2026年5月9日，你是一个助手..."}
  ]
}

// ✓ 正确：system prompt 保持稳定
{
  "messages": [
    {"role": "system", "content": "你是一个助手..."},
    {"role": "user", "content": "今天是2026年5月9日，请帮我..."}
  ]
}
```

将动态内容放在 messages 末尾（用户消息中），而非 system prompt 中，这样不影响前缀缓存。

四、Request 结构设计原则

合理的请求结构设计是提升缓存命中率的基础。

4.1 核心原则

- **Key 不变**：messages 中各消息的 role 保持稳定。
- **Key 的个数不变**：消息数量结构保持一致。
- **Key 的顺序不变**：消息排列顺序保持一致。

4.2 Message 结构变化原则

末尾追加 Key：新的对话轮次只在 messages 数组末尾追加，不要在中间插入或修改已有消息。

```
// 第 1 轮
{
  "messages": [
    {"role": "system", "content": "你是助手"},
    {"role": "user", "content": "问题1"}
  ]
}
```

```
// 第 2 轮（末尾追加，前缀不变）
{
  "messages": [
    {"role": "system", "content": "你是助手"},
    {"role": "user", "content": "问题1"},
    {"role": "assistant", "content": "回答1"},
    {"role": "user", "content": "问题2"}
  ]
}
```

这样前两条消息的 KV Cache 可以被完全复用。

五、新版本发版建议

在产品发版更新时，如果涉及 System Prompt 变更，可能导致缓存大面积失效。建议：

1. **发版前预热缓存**：少量模拟会话数据访问 API，提前构建 KV Cache。
2. **避免突增流量冲击**：灰度发布，防止发版 + 突增流量同时命中导致 Cache Rate 骤降。
3. **监控 Cache Rate**：发版后密切关注缓存命中率指标，如异常下降及时排查。

六、总结

优化手段	作用	使用方式
`prompt_cache_key`	标识相同上下文，提升缓存复用	请求体字段，值为 conversation_id
`X-Session-ID`	路由到同一实例，提升局部命中	HTTP Header
稳定 System Prompt	避免缓存因前缀变化而失效	不写入动态时间内容
末尾追加消息	保持前缀一致性	messages 只在末尾追加
发版前预热	防止冷启动冲击	提前模拟请求构建 KV