【对齐OPENAI】chat/completions模型对话接口 (支持OpenAI、Gemini、VenusLLMServing等)

2025-12-15 11:53

Table of Contents

需要特别注意：

1. **外部模型（OpenAI、Claude等）需要申请权限**：
   【模型权限】OpenAI、Claude、Gemini等闭源LLM使用权限申请
2. **token 绑定应用组进行计费。计费说明：** 【成本】模型权限及成本结算
3. 外部模型权限申请后、公共模型调用 请使用modelid进行调用：
   【模型列表】支持API调用的公共模型及外部模型

URL: `http://v2.open.venus.oa.com/llmproxy/chat/completions`

Type: POST

Author: christruan

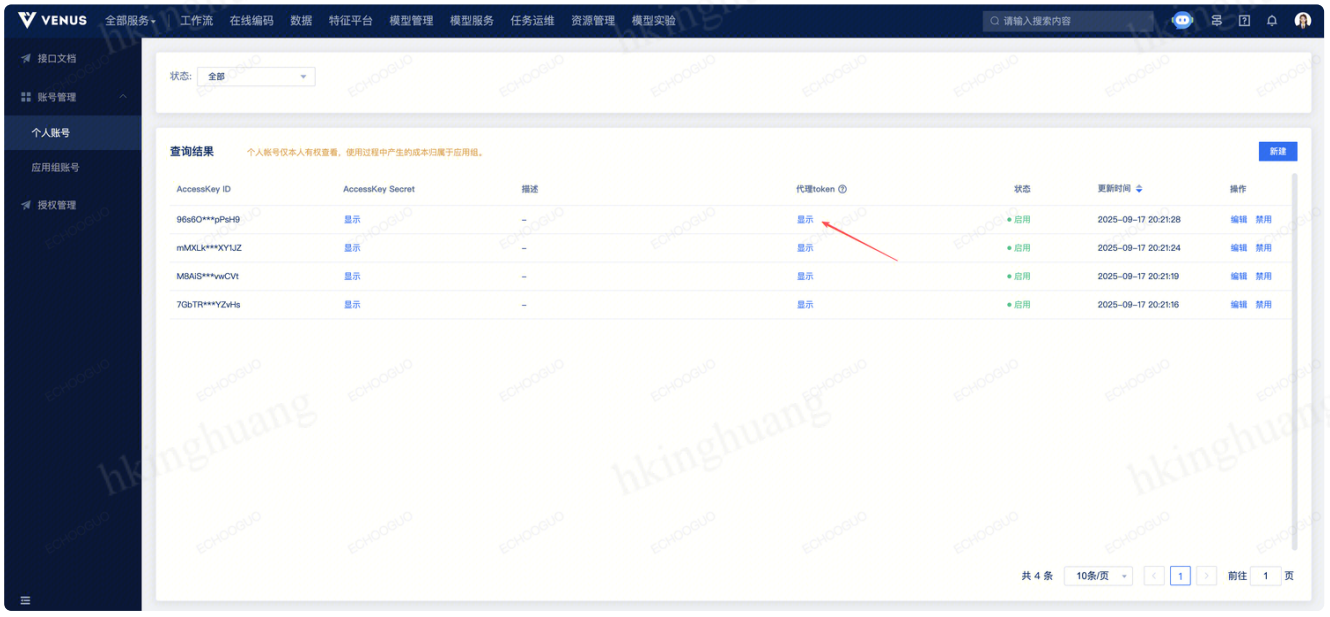
Content-Type: application/json; charset=utf-8

Description: 完全对齐 OpenAI （流式&非流式）请求和响应，支持 Venus 上所有外部模型、公共模型、私有服务

一、请求

0、生成代理token：

(1)进入Venus个人账号https://venus.woa.com/#/openapi/accountManage/personalAccount 或 应用组账号 https://venus.woa.com/#/openapi/accountManage/appgroupAccount：



❓个人帐号和应用组账号的区别：

- 使用过程中的成本均归属于应用组；
- 个人帐号：拥有正式应用组的所有人均可创建，仅本人有权查看，在人员变动时需作交接处理；
- 应用组账号：仅管理员可创建，应用组内所有成员可见隐藏敏感信息的账号；管理员可为每个应用组账号配置相应的维护人，维护人有权查看完整的账号信息。

（2）选择代理应用组，生成token后使用（注意，请选择合适的正式应用组，应用组 与成本、模型权限相关）

1. 简单请求

Body-parameters:

Parameter	Type	Description	Required
model	string	1. 平台公共模型 (查询所有支持的模型列表，通过该接口可以查询可调用的所有平台公共模型列表，取其中的 model 字段，注意平台公共模型并发有限，若高并发请应用组内部署服务) 2. 私有化服务，格式为 `server:212345` ，其中 `212345` 是 Venus 的 serverId 	true
messages	array	消息列表，自定义上下文	true
temperature	float	使用什么取样温度，0到2之间。较高的值(如0.8)将使输出更加随机，而较低的值(如0.2)将使输出更加集中和确定。	false
top_p	float	温度抽样的另一种选择，称为核抽样，其中模型考虑具有top_p概率质量的令牌的结果。所以0.1意味着只考虑包含前10%概率质量的标记。	false
top_k	int32	与top_p类似，它也是通过设置一个概率阈值来限制模型生成的候选词的数量。不同的是，Top_k是根据候选词的概率进行选择，只选择概率最高的K个候选词。	false
max_tokens	int32	最大模型输出 tokens 数	false（某些模型必填，比如 Claude 系列）

Request-example:

```
{
    // model 可以切换为任意平台支持的外部模型、公共模型、私有服务
    // 如果调用私有服务, model 需要被设置为 "server:212345", 其中 `212345` 是 serverId
    "model": "gpt-3.5-turbo",
    "messages": [
        // 可用的 role 有:
        // 1. system - system prompt
        // 2. user - 用户 prompt
        // 3. assistant - 模型响应
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "hello"},
        {"role": "assistant", "content": "Hello! How can I help you today?"},
        {"role": "user", "content": "who are you?"}
    ],
    "max_tokens": 1000,
    "temperature": 0.1
}
```

2. 高级参数

2.1 OpenAI 高级参数

更多支持的参数和解释见：API Reference - OpenAI API

Body-parameters:

Parameter	Type	Description	Required
tools	array	模型可以调用的工具列表 注意事项： 1. `tools.function.name` 字段必须是 a-z、a-z、0-9 或包含下划线和破折号，最大长度为 64。不符合规范的`function.name`将有可能导致`tool_calls`解析失败	false
tool_choice	string or object	控制模型调用哪个函数(如果有的话) 注意事项： 1. kimi-k2模型不支持 `tool_choice=required` 参数，官方说明文档：https://platform.moonshot.cn/docs/guide/migrating-from-openai-to-kimi#%E5%85%B3%E4%BA%8E-tool_choice	false
presence_penalty	float	-2.0到2.0之间的数字	false
frequency_penalty	float	-2.0到2.0之间的数字	false
seed	int32	请求的种子。当保持所有其他参数一致的情况下，使用相同的 seed，模型的输出会保持固定。仅 `gpt-3.5-turbo-1106`、`gpt-4-1106-preview`、 `gpt-4-vision-preview` 可用	false
response_format	object	指定模型输出的结构(又称JSON模式），让模型严格按照 JSON 格式来输出，以实现输出的结构化，便于后续逻辑进行解析。公共API 中仅 GPT系列、Gemini系列所有模型、`deepseek-v3-local-II`、`deepseek-r1-local-II`、`deepseek-r1-local-III`可用。 使用demo请参考：https://api-docs.deepseek.com/zh-cn/guides/json_mode 注意事项： 1. 设置 `response_format` 参数为 `{'type': 'json_object'}`或`{'type': 'json_schema'}` 2. 用户传入的 system 或 user prompt 中必须含有 `json` 字样，并给出希望模型输出的 JSON 格式的样例，以指导模型来输出合法 JSON。 3. 需要合理设置 `max_tokens` 参数，防止 JSON 字符串被中途截断。	false
└─ type	string	`json_object` ，当设置 `response_format` 参数为 `{'type': 'json_object'}`。表示要求模型返回一个JSON对象，以JSON字符串的格式返回。 `json_schema`，用于约束JSON的输出字段，类型。当设置 `response_format` 参数为 `{'type': 'json_schema'}`时，模型会按用户提供的JSON Schema生成指定的JSON响应。	false

└─ json_schema	object	当 `type = "json_schema"` 时，用户必须提供符合规范的JSON Schema，用于自定义模型输出的JSON格式。关于JSON_Schema格式的介绍请见链接	false
reasoning_	stri	限制推理模型的思考力度。当前支持的选项为 `low`、`medium`、`high`	false

2.2 Claude 高级参数

Body-parameters:

Parameter	Type	Description	Required
thinking_enabled	boolean	是否开启思考模式，仅对支持思考模型的模型生效，非思考模型请不要传入该参数！	false
thinking_tokens	int32	控制思考的长度，该值必须大于 1024 且不超过 `max_tokens`。仅对支持思考模型的模型生效，非思考模型请不要传入该参数！	false

```
{
  "model": "claude-3-7-sonnet-20250219",
  "messages": [
    {
      "role": "user",
      "content": "你好"
    }
  ],
  "stream": true,
  // 开启 thinking mode
  "thinking_enabled": true,
  // 如果开启 thinking mode, thinking_tokens 必传, 且不能大于 max_tokens
  "thinking_tokens": 2048,
  "max_tokens": 4096
}
```

2.2.1 Prompt Cache

Claude 支持开启 Prompt Cache，可以用于 `tools` / `system` / `messages` 上，具体指导见 Anthropic 官方文档。

需要注意：

1. 缓存长度最小为 `1024` ，请确保缓存的区间的 tokens 超过该值，否则缓存不生效。
2. 为了提升整体吞吐，底层会集合多个服务商。当有启用 Prompt Cache 需求时，可以在请求 header 中添加请求头 Venus-Sticky-Routing 开启粘性路由，可选的有：
   a. `Venus-Sticky-Routing: token` ，表示按照请求的 Bearer token 粘性路由。
   b. `Venus-Sticky-Routing: trace` ，表示按照链路追踪的 trace_id 粘性路由。trace 应该要符合 OpenTelemetry 协议，或者在请求头中手动添加 `'traceparent': "00-{trace_id_hex}-{span_id_hex}-01"` ，其中 trace_id 应该是 32 位长 16 进制，span_id 应该是 16 位长 16 进制。
3. 开启粘性路由后，因为短时间内会持续路由到一个底层资源，相应的并发能力会受影响，请注意做好重试。

以下是一个请求 demo：

```jsonc
{
  "model": "claude-3-7-sonnet-20250219",
  "stream": true,
  "max_tokens": 10240,
  "temperature": 0.1,
  "messages": [
    {
      "role": "system",
      "content": [
        {
          "type": "text",
          "text": "<your system prompt here>",
          // 对 system prompt 部分打上 cache 标签
          // 如果你的多个请求的 system prompt 一样，仅第一次请求会消耗 system_prompt_tokens * prompt_tokens_price * 1.25 的费用
          // 后续所有相同的 system prompt 请求只消耗 system_prompt_tokens * prompt_tokens_price * 0.1 的费用
          "cache_control": {
            // type: ephemeral 会缓存该 prompt 5 分钟
            "type": "ephemeral"
          }
        }
      ]
    },
    {
      "role": "user",
      "content": "<Round 1 User Prompt>"
    },
    {
      "role": "assistant",
      "content": "<Round 1 Assistant Response>",
      "tool_calls": [
        {
          "id": "toolu_xxx",
          "type": "function",
          "function": {
            "name": "random_tool",
            "arguments": "{\"a\": \"b\"}"
          }
        }
      ]
    },
    {
      "role": "tool",
      "content": [
        {
          "type": "text",
          "text": "<Round 1 Tool Response>",
          // 在此处添加 cache_control 字段，会检查 “非 system prompt 的最开始的 message” 到 “该 message” 的所有内容是否已经缓存
          // 如果未检查到缓存，会将 “非 system prompt 的最开始的 message” 到 “该 message” 的所有内容写入缓存（这一步会产生 1.25 倍的 prompt 费用）
```

```
            // 如果检查到缓存，那么 “非 system prompt 的最开始的 message” 到 “该 message” 的所有内容将从缓存中读取（这一部会产生 0.1 倍的 prompt 费用）
            "cache_control": {
              "type": "ephemeral"
            }
          }
        ],
        "name": "random_tool",
        "tool_call_id": "toolu_xxx"
      },
      {
        "role": "user",
        "content": [
          {
            "type": "text",
            "text": "<Round 2 User Prompt>"
            // 在此处添加 cache_control 字段，意味着将 context 中的上一个未被缓存的 message 到当前 message 的部分写入缓存
            // 理想情况下, 你的 messages 部分应该这么组成：
            // <system_prompt> <cache_control> <already cached context> <cache_control> <uncached context> <cache_control>
            "cache_control": {
              "type": "ephemeral"
            }
          }
        ]
      },
    ],
    "tool_choice": "auto",
    "tools": [
      {
        "type": "function",
        "function": {
          "name": "random_tool",
          "description": "random tool description",
          "parameters": {
            "type": "object",
            "properties": {
            },
            "required": []
          }
        }
      },
      {
        "type": "function",
        "function": {
          "name": "another_random_tool",
          "description": "another random tool description",
          "parameters": {
            "type": "object",
            "properties": {
            },
            "required": []
```

```
        },
        // 在 tools 的最后一个 tool 中添加 cache_control 字段，可以缓存所有以上的 tools 定义
        "cache_control": {
          "type": "ephemeral"
        }
      }
    }
  ]
}
```

Prompt Cache Response example：

```
{
    "id": "msg_vrtx_01FwrvqnS6LA2bPbj4wXpHCB",
    "object": "chat.completion",
    "created": 1754480720,
    "model": "claude-3-7-sonnet-20250219",
    "choices": [
        {
            "message": {
                "role": "assistant",
                "content": "<Assistant Response>",
                "tool_calls": [
                    {
                        "id": "toolu_vrtx_01SeEYzXrBURrabgsnSCXkoS",
                        "type": "function",
                        "function": {
                            "name": "random_tool",
                            "arguments": "{}"
                        }
                    }
                ]
            },
            "finish_reason": "tool_calls"
        }
    ],
    "usage": {
        // 本次请求的完整 prompt tokens，即：uncached_prompt_tokens + cache_read_tokens + cache_creation_tokens
        "prompt_tokens": 25194,
        "completion_tokens": 443,
        "total_tokens": 25637,
        "prompt_tokens_details": {
            // 击中缓存的 tokens 数
            "cache_read_tokens": 22653,
            // 写入缓存的 tokens 数
            "cache_creation_tokens": 586
        }
        // 最终计价为：
        // (prompt_tokens - cache_read_tokens - cache_creation_tokens) * prompt_tokens_price
        // + completion_tokens * completion_tokens_price
        // + cache_read_tokens * prompt_tokens_price * 0.1
```

```
        // + cache_creation_tokens * prompt_tokens_price * 1.25
    },
    "venusMarker": {
        "spanId": "2d0fafffb4b7fb45"
    }
}
```

2.3 Gemini 高级参数

Body-parameters:

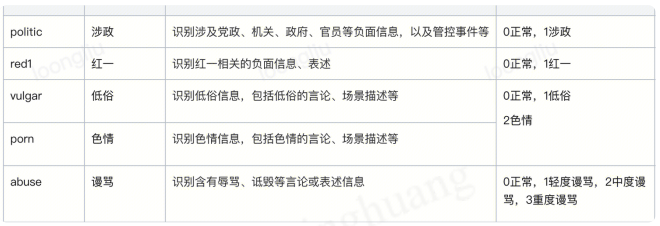
Parameter	Type	Description	Required
presence_penalty	float	-2.0到2.0之间的数字	false
frequency_penalty	float	-2.0到2.0之间的数字	false
thinking_enabled	boolean	是否开启思考模式，仅对支持思考模型的模型生效，非思考模型请不要传入该参数！ 目前仅对 `gemini-2.5-flash` / `gemini-2.5-flash-lite` 生效，请一定主动设置开启或关闭，否则由模型自主决定，不可控。	false
thinking_tokens	int32	控制思考的长度	false
google_search_enabled	boolean	是否启用 Google Search，默认关闭，单次搜索成本 $0.035。仅支持 gemini 2.0 以上的系列。	false
image_config	object	图片配置	false
└─ aspect_ratio	string	图片比例，可选：`"1:1"`, `"3:2"`, `"2:3"`, `"3:4"`, `"4:3"`, `"4:5"`, `"5:4"`, `"9:16"`, `"16:9"`, `"21:9"`	false
└─ image_size	string	图片大小，可选：`"1K"`, `"2K"`, `"4K"`。	false
response_format	object	指定模型输出的结构(又称JSON模式），让模型严格按照 JSON 格式来输出，以实现输出的结构化，便于后续逻辑进行解析。公共API 中仅 GPT系列、Gemini系列所有模型、`deepseek-v3-local-II`、`deepseek-r1-local-II`、`deepseek-r1-local-III` 可用。 使用demo请参考：https://api-docs.deepseek.com/zh-cn/guides/json_mode 注意事项： 1. 设置 `response_format` 参数为 `{'type': 'json_objec t'}` 或 `{'type': 'json_schema'}` 2. 用户传入的 system 或 user prompt 中必须含有 `jso n` 字样，并给出希望模型输出的 JSON 格式的样例，以指导模型来输出合法 JSON。 3. 需要合理设置 `max_tokens` 参数，防止 JSON 字符串被中途截断。	false
thinking_level	string	`gemini-3-pro` 支持，用于指定模型在生成响应时的推理预算。通过选择两种状态之一，可以在响应质量与推理复杂度、延迟及成本之间进行权衡。 • `LOW`: 最小化延迟和成本。适用于指令遵循或对话场景。	false

		• `HIGH`: 最大化推理深度。默认值。动态思考模式。模型可能需要更长时间才能生成首个 token，但输出内容会经过更充分的推敲验证。	
media_resolution	string	`gemini-3-pro` 支持，用于对多模态视觉处理进行精细化控制。更高的分辨率能提升模型读取精细文本或识别细节的能力，但会增加 token 用量和延迟。 media_resolution 参数决定了为每张输入图像、PDF 页面或视频帧分配的最大 token 数量。可选： • `MEDIA_RESOLUTION_LOW` • `MEDIA_RESOLUTION_MEDIUM`	false

```
{
  "model": "gemini-2.5-flash",
  "messages": [
    {
      "role": "user",
      "content": "你好"
    }
  ],
  "stream": true,
  // 图片配置, 仅 Gemini 的生图模型有效
  "image_config": {
    "aspect_ratio": "1:1"
  },
  // 开启 thinking mode
  "thinking_enabled": true,
  // 如果开启 thinking mode, thinking_tokens 必传, 且不能大于 max_tokens
  "thinking_tokens": 2048,
  "max_tokens": 4096
}
```

2.4 VenusLLMServing 高级参数（Venus平台部署的模型服务）

Body-parameters:

Parameter	Type	Description	Required
thinking_enabled	boolean	是否开启思考模式，仅对支持思考模型的模型生效，非思考模型请不要传入该参数！ 目前仅支持 Qwen3 系列、deepseek-v3.1-terminus、glm-4.5、deepseek-v3.2模型	false
stream_options	object	流式输出的选项	false
└─ include_usage	boolean	流式输出时，是否需要在最后一个 chunk 返回 usage，此时 choices 会是 []	false
response_format	object	指定模型输出的结构,(又称JSON模式）。目前仅支持SGLang引擎服务，平台部署的公共服务仅 `deepseek-v3-local-II`、`deepseek-r1-local-II`、`deepseek-r1-local-III`可用。 注意事项： 1. 设置 `response_format` 参数为 `{'type': 'json_object'}` 或`{'type': 'json_schema'}` 2. 用户传入的 system 或 user prompt 中必须含有 `json` 字样，并给出希望模型输出的 JSON 格式的样例，以指导模型来输出合法 JSON。 3. 需要合理设置 `max_tokens` 参数，防止 JSON 字符串被中途截断。	false
└─ type	string	`json_object` ，当设置 `response_format` 参数为 `{'type': 'json_object'}`。表示要求模型返回一个JSON对象，以JSON字符串的格式返回。 `json_schema`，用于约束JSON的输出字段，类型。当设置 `response_format` 参数为 `{'type': 'json_schema'}`时，模型会按用户提供的JSON Schema生成指定的JSON响应。	false
└─ json_schema	object	当 `type = "json_schema"` 时，用户必须提供符合规范的JSON Schema，用于自定义模型输出的JSON格式；关于JSON_Schema的介绍请见链接	false
risk_control	object	风控审核相关参数，注：风控当前无法给业务风险做兜底，上线前如需兜底请联系marshao或VenusHelper 进行风控策略定制 和 对接沟通 	false
└─ enable	bool	是否开启风控审核	true
└─ platform	string	风控审核平台，当前可选: safety_guard	true

politic	涉政	识别涉及党政、机关、政府、官员等负面信息，以及管控事件等	0正常，1涉政
red1	红一	识别红一相关的负面信息、表述	0正常，1红一
vulgar	低俗	识别低俗信息，包括低俗的言论、场景描述等	0正常，1低俗 2色情
porn	色情	识别色情信息，包括色情的言论、场景描述等	
abuse	谩骂	识别含有辱骂、诋毁等言论或表述信息	0正常，1轻度谩骂，2中度谩骂，3重度谩骂

└─ check_character_interval	int	流式输出的场景下，执行风控审核所间隔的字符数，间隔字符数越小，对输出耗时影响越大。默认间隔5个字符	false
logprobs	bool	是否返回输出标记的对数概率。如果为 true，则返回消息内容中返回的每个输出标记的对数概率	false
top_logprobs	int	0 到 20 之间的整数，指定每个标记位置最有可能返回的标记数量，每个标记都有关联的对数概率。如果使用此参数，logprobs 必须设置为 true	false

```
{
    # 自定义JSON Schema
    json_schema = json.dumps(
        {
            "type": "object",
            "properties": {
                "name": {"type": "string", "pattern": "^[\\w]+$"},
                "population": {"type": "integer"},
            },
            "required": ["name", "population"],
        }
    )
  "model": "qwen3-235b-a22b-fp8-local-II",
  "messages": [
    {
      "role": "user",
      "content": "Please generate the information of the capital of France in the JSON format."
    }
  ],
  "response_format": {
    "type": "json_schema",
    "json_schema": {
      "name": "capital_info",
      "schema": json.loads(json_schema) #提前定义好的json_schema
    }
  },
  "stream": true,
  // 开启 thinking mode
  "thinking_enabled": true,
  "max_tokens": 4096,
  // 风控审核相关配置，确认需要才开启
  "risk_control": {
    "enable": false,
    "platform": "safety_guard"
  }
}
```

2.5 多模态

注意事项：

1. 使用多模态能力的时候，不同于文本类型，`$.messages.content` 是 `array` 类型。

2. `$.messages.content` 支持按照 OpenAI 官方提供的 `type`:
    a. **text**
    b. **image_url**
3. Venus 额外提供了存储桶，方便用户上传自己的多模态文件，由 Venus 对文件生成公网可访问的 url（代码示例见： 将多模态文件上传至 Venus COS / Google Cloud Storage ）。此时需要将 type 变更为 `venus_image_url` / `venus_multimodal_url` / `gemini_multimodal_url`。
4. detail 参数仅支持 OpenAI 模型。`$.messages.content.detail` 为 high 时，是 low 的 10~12 倍。一张 high 的图片大概消耗 765 ~ 1105 tokens；一张 low 的图片大概消耗 85 tokens。
5. 注意：只有特定的 多模态模型才能支持 （gpt 系列、gemini 系列、claude 系列 多模态模型部署的服务等）
6. 私有化部署的VIlm服务 默认单次请求最多传10张图片

```
{
  "model": "gpt-4-vision-preview",
  "messages": [
    {
      "role": "user",
      "content": [
        {
          "type": "text",
          "text": "解释这三张图片的区别"
        },
        // OpenAI 官方提供的图片 url 形式
        // 仅 OpenAI 和 Venus 公共模型支持直传 http url
        // 请注意提交给 OpenAI 的 url 确保外网可访问
        {
          "type": "image_url",
          "image_url": {
            // 仅 OpenAI 多模态模型支持 detail 参数
            "detail": "high",
            "url": "https://xxxxx/xx.png"
          }
        },
        // 图片 base64 编码形式, 所有模型服务均可用
        {
          "type": "image_url",
          "image_url": {
            "url":
"....CC="
          }
        },
        // Venus 提供的存储桶, 上传多模态文件 demo 见:
https://iwiki.woa.com/p/4009144950
        {
          "type": "venus_multimodal_url",
          "venus_multimodal_url": {
            // 必填, 多模态类型
            "mimeType": "video/mp4",
            "bucketName": "venus-llm-1258344701",
            "filepath": "chat-agency/attachments/rtx/xxx/c1e7bbd1-a94e-4f75-
aac3-bbff296f7b73-test.mp4"
          }
        },
```

```
        // 使用 vrn (venus 资源 id)，当把 COS 资源注册到 venus 时可用. 可在 https://venus.woa.com/#/subsystem/resource/appGroupResource/COS 查看当前应用组可用的存储桶.
        {
          "type": "venus_multimodal_url",
          "venus_multimodal_url": {
            "resId": 123,
            // 必填, 多模态类型
            "mimeType": "video/mp4",
            "filepath": "test.mp4"
          }
        },
        // Venus 多模态, 使用 base64 编码或者 url，超过10M的文件，请用上文COS的方式调用，否则会因为请求体过大失败
        {
          "type": "venus_multimodal_url",
          "venus_multimodal_url": {
            // 必填, 多模态类型，支持的 mimeType 有：
            // 图片类型：image/png，image/jpeg，image/jpg
            // 视频类型：video/mov，video/mp4
            // 音频类型：audio/wav，audio/mp3，audio/mpeg
            "mimeType": "video/mp4",
            "url": "data:video/mp4;base64,xxxxxxx"
            // 或者可以使用 url, 请注意该 url 对于目标模型是否可调用. 比如 OpenAI 无法访问内网 url, 内网 VenusLLMServing 无法访问外网 url.
            // "url": "https://xxxx.com/xxx.png"
          }
        },
        // 当使用 Gemini 时，请使用 gemini_multimodal_url。如果文件较大（大于 4MB），可以将文件上传至 GoogleStorage，见 https://iwiki.woa.com/p/4009144950
        {
          "type": "gemini_multimodal_url",
          "gemini_multimodal_url": {
            // Gemini 支持的 mimeType 有：
            // 文本类型：text/plain
            // 图片类型：image/png，image/jpeg
            // 视频类型：video/mov，video/mpeg，video/mp4，video/mpg，video/avi，video/wmv，video/flv
            // 文档类型：application/pdf
            // 音频类型：audio/wav，audio/mp3，audio/mpeg
            "mimeType": "video/mp4",
            "bucketName": "venus-gcp-aigc",
            "filepath": "chat-agency/attachments/rtx/SINGLE_CHAT/c1e7bbd1-a94e-4f75-aac3-bbff296f7b73-test.mp4"
          }
        },
        // 当使用 Gemini 时，请使用 gemini_multimodal_url。如果文件较小（小于 4MB），也可以使用 base64 编码
        {
          "type": "gemini_multimodal_url",
          "gemini_multimodal_url": {
            "mimeType": "video/mp4",
            "encoded": "iVBORw0KGgoAAAANSUhEUgAABAAAAA....CC="
```

```
          }
        }
      ]
    }
  ],
  "max_tokens": 1000
}
```

3. 开启流式输出

在上述参数的基础上，可以通过 `stream` 参数来控制是否开启流式输出。默认关闭，设置为 `true` 时将开启流式输出。

Parameter	Type	Description	Required
stream	boolean	是否开启流式输出，默认关闭	false

Request-example:

```
{
  "model": "gpt-3.5-turbo",
  "messages": [
        {"role": "user", "content": "hello"}
  ],
  "stream": true
}
```

开启流式输出后，所有模型和服务都将按照 OpenAI 标准流式响应结构返回

3.2 代码示例

先安装 Python 的 sse 客户端: `pip install -U sseclient-py==1.7.2`

```
import json
import requests
import sseclient

token = "<your token>" ### 获取代理token，注意应用组选择正确 https://venus.woa.com/#/openapi/accountManage/personalAccount

url = "http://v2.open.venus.oa.com/llmproxy/chat/completions"

payload = {
    'model': 'gpt-3.5-turbo',
    'stream': True,
    'messages': [
        {
            'role': 'system',
            'content': 'You are a helpful assistant.'
        },
        {
            'role': 'user',
            'content': 'Hello'
```

```
        }
    ],
    'max_tokens': 500
}

headers = {
    'Content-Type': 'application/json',
    'Authorization': f'Bearer {token}'
}

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

# 判断是否异常
if response.status_code != 200:
    print(response.json())
    exit()

# 建立 SSE 连接, 将 response 用 sseClient 包装
client = sseclient.SSEClient(response)
for event in client.events():
    print(event.data)
    print()

```

二、响应

所有模型和服务的响应结构会对齐 OpenAI 的响应结构，**不排除有某些字段的缺失**。

1. 正常响应

1.1 OpenAI 响应示例

```
{
    "id": "chatcmpl-xxx",
    "object": "chat.completion",
    "created": 1710229978,
    "model": "gpt-35-turbo",
    "choices": [
        {
            "index": 0,
            // 流式输出时, message 字段为 delta；本身是顺序标记,当请求参数 `n` 大于1时（例如 `n=3`），API会一次性返回多个生成的文本结果（即多个`choice`）
            "message": {
                "role": "assistant",
                "content": "你好！有什么我可以帮助你的吗？"
            },
            "finish_reason": "stop"
          //大型语言模型(LM)的对话接口中，`finish_reason` 字段用于表示模型生成回复停止的原因。以下是常见的枚举值及其含义：

```

```
//| 枚举值 | 含义 | 说明 |
//|--------|------|------|
//| `stop` | 正常停止 | 模型自然完成了回复，如达到了一个指定的停止词(如"\n\n")或自然结束 |
//| `length` | 长度限制 | 回复达到了最大令牌限制(max_tokens)，被截断 |
//| `content_filter` | 内容过滤 | 由于内容安全策略或敏感内容过滤而中止生成 |
//| `function_call` | 函数调用 | 模型停止生成以调用一个指定的函数(用于工具调用) |
//| `tool_calls` | 工具调用 | 模型停止生成以调用一个工具(较新的接口中的用法) |
//| `null` 或 `None` | 未完成 | 流式输出中尚未完成的消息，或处理异常 |
//| `cancelled` | 被取消 | 请求被客户端或服务器中断/取消 |
//| `recitation` 或 `content_policy` | 内容政策违规 | 检测到抄袭、违规内容或不适当内容 |
//| `error` | 错误 | 处理请求时出现错误 |

//## 不同模型的特殊枚举值

//不同的LLM服务可能有各自特殊的枚举值：

//### OpenAI (ChatGPT/GPT系列)
//- 基础枚举: `stop`, `length`, `content_filter`, `function_call`, `tool_calls`
//- 特殊枚举: `null` (流式响应中), `content_policy` (政策违规)

//### Anthropic (Claude系列)
//- 基础枚举: `stop`, `max_tokens`, `stop_sequence`
//- 特殊枚举: `cancelled_by_user`, `cancelled_by_system`

//### Google (Gemini/PaLM系列)
//- 基础枚举: `stop`, `max_tokens`, `safety`
//- 特殊枚举: `recitation` (识别抄袭), `other`

//### Azure OpenAI
//- 基本与OpenAI相同
//- 在流式API中，未完成的中间响应会使用 `null` 表示

//## 在应用中处理 finish_reason

//根据 finish_reason 的不同值，可以采取不同的处理策略：

//- `stop`: 正常处理完整回复
//- `length`: 可能需要提示用户回复被截断，或请求继续生成
//- `content_filter`/`content_policy`: 可以向用户解释内容被过滤的原因
//- `function_call`/`tool_calls`: 执行相应的函数调用并继续对话
//- `null`: 在流式接口中等待完整回复
//- `error`: 实施错误处理和重试机制

//在开发应用时，建议针对所有可能的 finish_reason 值设计相应的处理逻辑，以提供最佳的用户体验。
        }
    ],
    // usage 不保证一定返回
    "usage": {
        "prompt_tokens": 9,
        "completion_tokens": 18,
        "total_tokens": 27
    },
    // venus 的附带标记, 业务调用时最好能够记录下其中的 spanId, 方便追踪
```

```
    "venusMarker": {
        "spanId": "dd499dc32bc772d2"
    }
}
```

1.2 Claude 响应示例

```
{
    "id": "msg_01XMwiYJEqC4ZczL1ezgsn8u",
    "model": "claude-3-sonnet-20240229",
    "choices": [
        {
            // 流式输出时, message 字段为 delta
            "message": {
                "role": "assistant",
                "content": "您好!我是人工智能助手Claude,很高兴为您提供帮助。有什么我可以协助您的吗?"
            }
        }
    ],
    // usage 不保证一定返回
    "usage": {
        "prompt_tokens": 10,
        "completion_tokens": 42,
        "total_tokens": 52
    },
    // venus 的附带标记, 业务调用时最好能够记录下其中的 spanId, 方便追踪
    "venusMarker": {
        "spanId": "44dcfba30043fbc7"
    }
}
```

1.3 Gemini 响应示例

- 支持 `gemini-2.5-flash-image、gemini-3-pro-image` 图片的返回
- 支持 `gemini-3-pro` 的 tool call

```
{
    "object": "chat.completion",
    "created": 1757496455,
    "model": "gemini-2.5-flash-image",
    "choices": [
        {
            "index": 0,
            // 流式输出时, message 字段为 delta
            "message": {
                "role": "assistant",
                // 从 gemini-2.5-pro 起，模型返回的 tool_calls 中会携带 thought_signature
                // 在 gemini-3-pro 中，该参数需要强制传回
                // 具体说明见：https://docs.cloud.google.com/vertex-ai/generative-ai/docs/start/get-started-with-gemini-3#thought_signatures
                "tool_calls": [
```

```
                    {
                        "index": 1,
                        "id": "6f935223-a389-4321-8594-ab4481d83edc",
                        "function": {
                            "name": "get_temperature",
                            "arguments": "{\"location\":\"shenzhen\"}"
                        },
                        "thought_signature": "CtACAePx..."
                    },
                    {
                        "index": 2,
                        "id": "7bdc8425-13ee-4aff-ba77-4fc690de5c97",
                        "function": {
                            "name": "get_humidness",
                            "arguments": "{\"location\":\"shenzhen\"}"
                        }
                    }
                ]
                // 由于 Gemini 可能返回多模态内容, 因此 content 有可能为数组结构 ，例如 gemini-2.5-flash-image 时会返回图片
                // 当前 Content 的 type 有：text | venus_multimocal_url
                "content": [
                    {
                        "type": "text",
                        "text": "Here's that poster for you! "
                    },
                    {
                        "type": "venus_multimodal_url",
                        "venus_multimodal_url": {
                            "mimeType": "image/png",
                            "url": "...="
                        }
                    }
                ]
            },
            "finish_reason": "STOP"
        }
    ],
    "usage": {
        "prompt_tokens": 46,
        "completion_tokens": 1299,
        "total_tokens": 1345
    },
    "venusMarker": {
        "spanId": "4cc081429ed923ca"
    }
}
```

1.4 思维链模型响应示例

```
{
    "id": "msg_01XMwiYJEqC4ZczL1ezgsn8u",
    "model": "deepseek-reasoner",
```

```
    "choices": [
        {
            // 流式输出时, message 字段为 delta
            "message": {
                "role": "assistant",
                "content": "你好！很高兴见到你，有什么我可以帮忙的吗？无论是问题、建议还是闲聊，我都在这里为你服务。😊",
                // 对于有思维链过程的模型, 思考过程会放在 reasoning_content 字段
                "reasoning_content": "你好！很高兴见到你，有什么我可以帮忙的吗？无论是问题、建议还是闲聊，我都在这里为你服务。😊"
            }
        }
    ],
    // usage 不保证一定返回
    "usage": {
        "prompt_tokens": 10,
        "completion_tokens": 42,
        "total_tokens": 52
    },
    // venus 的附带标记, 业务调用时最好能够记录下其中的 spanId, 方便追踪
    "venusMarker": {
        "spanId": "44dcfba30043fbc7"
    }
}
```

2. 异常响应

当请求过程中有异常，不论是否开启流式，该请求会返回 HttpCode 非 200，并且 body 中附带异常信息。

Response-example:

```
{
    "error": {
        "message": "max_tokens: must be greater than or equal to 1",
        "type": "invalid_request_error"
    },
    "venusMarker": {
        "spanId": "c03d912888e3da97"
    }
}
```

三、代码示例

1. Http 客户端

```
import json
import requests

token = "<your token>"

```

```
url = "http://v2.open.venus.oa.com/llmproxy/chat/completions"

payload = {
    'model': 'gpt-3.5-turbo',
    'messages': [
        {
            'role': 'system',
            'content': 'You are a helpful assistant.'
        },
        {
            'role': 'user',
            'content': 'Hello'
        }
    ],
    'max_tokens': 500
}

headers = {
    'Content-Type': 'application/json',
    'Authorization': f'Bearer {token}'
}

response = requests.post(url, headers=headers, data=json.dumps(payload))

# 判断是否异常
if response.status_code != 200:
    # 异常处理
    print(response.json())
    exit()

print(response.json())

```

2. OpenAI 包

安装 OpenAI 包 `pip install openai==1.58.1 pydantic==2.10.4`

```
import os
from openai import OpenAI

os.environ['OPENAI_API_KEY'] = "<your token>"
client = OpenAI(base_url="http://v2.open.venus.oa.com/llmproxy")

response = client.chat.completions.create(
    model="claude-3-7-sonnet-20250219",
    stream=True,
    messages=[{"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "你好👋！，你是谁啊"}],
    max_tokens=2048,
    extra_body={
        "thinking_enabled": True,
        "thinking_tokens": 1024
    }
```

```
)

for chunk in response:
    for choice in chunk.choices:
        if choice.delta and 'reasoning_content' in choice.delta.model_extra:
            # 思考内容
            print(choice.delta.model_extra['reasoning_content'], end="")
        if choice.delta and choice.delta.content:
            # 模型输出
            print(choice.delta.content, end="")

```

3. LangChain 包

```
from langchain_openai import ChatOpenAI
from langchain_core.messages import HumanMessage, SystemMessage


token = '<your token>'
url = 'http://v2.open.venus.oa.com/llmproxy'
model = 'qwen-14b-chat'

llm = ChatOpenAI(api_key=token, base_url=url, model_name=model)
###若是Venus服务，model_name=server:212345

messages = [
    HumanMessage(
        content="你好👋！，你是谁啊"
    ),
]

print(llm.invoke(messages))
```