天翼云
State Cloud

星辰 MaaS 模型服务平台

用户操作指南

天翼云科技有限公司

目录

产品介绍

产品定义

星辰MaaS模型服务平台集成了模型体验、模型服务、数据管理、模型调优、模型管理、服务部署、模型量化等核心功能，为用户提供功能全面、安全可靠且具有价格优势的模型调用服务以及从数据准备、模型训练到服务上线的全流程支持。平台致力于降低AI开发门槛，使不具备深厚技术背景的用户也能快速完成模型的调优、部署与应用。

模型体验

提供多场景模型展示与试用，支持开发者在模型广场快速体验主流AI模型，体验中心可零门槛测试基础推理能力。

模型服务

主流大模型提供标准化API接口，支持高并发在线推理服务，内置调用量监控与延迟分析功能，保障企业级应用稳定高效运行。

数据管理

平台包括公共数据集和个人数据集模块，可以实现共享和私有数据集，支持数据集的检索、预览与创建，为用户提供便捷的数据资源，支撑模型训练与调优。

模型调优

提供零门槛调优功能，用户无需编写代码，仅需选择或上传数据集，即可自动完成模型微调，快速定制出贴合特定场景的专业模型。

我的模型

为用户提供个人模型管理空间，支持对自建模型进行增、删、改、查操作，实现模型的集中纳管与版本控制。

服务部署

支持模型一键部署为在线推理服务，提供服务地址供外部调用，并支持服务上下线和管理等功能。

模型量化

支持常见的量化工具，可对模型进行压缩优化，减小模型体积与显存占用，提升推理性能，满足低成本、高吞吐的部署需求。

产品优势

模型服务丰富

集成主流先进第三方大模型，支持文本、图像、音视频等多模态处理能力。

操作使用方便

统一接口封装模型能力，开发者可直接调用API，实现分钟级集成。

零门槛开发

通过自动化调优工具与可视化界面，用户无需编写代码即可完成模型训练与部署，极大降低AI开发门槛。

全流程闭环

从数据集准备、模型调优、模型管理到服务部署与量化优化，平台提供完整的一站式AI应用开发链路。

可审计与可追溯

平台提供操作日志功能，确保模型开发与部署过程安全可控、可追溯。

术语解释

模型微调

是指利用预先训练好的神经网络模型，并针对特定任务在相对较少量的监督数据上进行重新训练的技术。这种方法能够充分利用预训练模型在大型数据集上学到的通用特征和知识，从而加速在新任务上的训练过程，并通常能够取得较好的性能表现。

Token

在自然语言处理中，token 通常指的是将文本分割成的最小单位，比如词语、子词或字符。在调用模型推理服务时，会将输入内容进行分词（tokenize），转化为模型可以理解的 token ，经过模型处理后，同样输出 token，并转化为您需要的文本或者其他内容载体。而模型处理（包括输入、输出）的 token 数量会被作为模型推理服务用量的一个重要计量单位。由于不同模型采用的分词策略不同，同一段文本可能会被转化为不同数量的 token。

数据集

是机器学习或深度学习模型训练过程中的重要组成部分。训练数据集是一组已知输入和对应输出的数据，用于训练模型以学习从输入到输出的映射关系。构建合适训练集，通过模型调优可增强模型能力，提升预测效果。

模型量化

通过降低模型参数的精度（如从FP16转为INT4），减小模型体积与显存占用，提升推理速度的技术。

模型使用限制

通过体验中心或调用API使用模型时，每个模型在上下文长度、最大输出长度、并发数、最大TPM/RPM数等方面均有不同数值的限制，部分限制信息可参见模型广场模型卡片信息。

计费说明

模型推理计费规则

本文介绍星辰MaaS模型服务平台推理服务的计费方式。

语音大模型-按量后付费

模型名称	单位	单价	备注
超多方言实时语音识别	元/小时	3	不足1小时，按实际秒数计费； 举例：服务使用时长=40秒，计费=3*40/3600=0.03元
实时超自然语音合成-普通话版	元/万字	2.4	不足1万字，按实际字符数计费； 举例：服务使用字符数=1000，计费=2.4*1000/10000=0.24元

语义大模型-按量后付费

模型名称	单位	输入单价	输出单价
对话大模型	元/千Tokens	0.003	0.012
通用问答大模型-36B	元/千Tokens	0.002	0.01

注意

1. 输入和输出不足1千Token，按实际消耗Token数计费；
2. 按 token 后付费的计算公式：在线推理费用 = 输入单价 × 输入token量 + 输出单价 × 输出token；举例：调用对话大模型，输入tokens=200，输出tokens=3500，则计费=0.003*200/1000+0.012*3500/1000=0.0006+0.042=0.0426元

文生图-按量后付费

模型名称	单位	单价
文生图大模型	元/次	0.3

图生图-按量后付费

模型名称	单位	单价
图生图大模型	元/次	0.3

模型训推计费规则

本文介绍星辰MaaS模型服务平台模型训推服务的计费方式。

算力包

销售品	价格（元）	用途
1万点算力包	100	训练、推理、量化任务时抵扣
2万点算力包	200	训练、推理、量化任务时抵扣
5万点算力包	500	训练、推理、量化任务时抵扣
10万点算力包	900	训练、推理、量化任务时抵扣
20万点算力包	1600	训练、推理、量化任务时抵扣
50万点算力包	3500	训练、推理、量化任务时抵扣

说明

1. 获取与上限：每个主账户我们都会免费赠送 10,000 算力点。您也可以自行购买，单次购买上限为100份。
2. 长期有效：算力点自购买之日起 12个月内有效，请您在有效期内使用。
3. 过期处理：有效期结束后，未使用的算力点将会失效。
4. 欠费与抵扣：如果账户欠费，充值后系统将优先自动抵扣所欠的算力点。

欠费说明

按量后付费账单按5分钟结算，并进行实时扣款。如果您账户中的可用额度（含账户余额和代金券）小于待结算的账单，会被认为欠费，对应的模型资源将被冻结。

欠费时长	说明
欠费时长 ⩽ 5 分钟	您可以正常使用模型服务，天翼云将按照您的具体使用量出具账单。
欠费时长 > 5 分钟	平台会给您发送相关通知，并关停服务。账号下的所有相关模型服务将无法正常使用。

账单和发票

账单

- 用户可在费用中心查看账单概览、账单详情；其中账单概览包含近6个月消费汇总、账期应付金额、汇总图表和表格等3个部分的内容，客户可以通过账单概览了解账期内的账单整体情况。账单详情可以多维度展示客户账单的详细信息，并可导出两种格式（xlsx、csv）的账单，在“导出记录”下载。
- 查看路径：进入“费用中心>账单管理> 账单概览 ”页面

发票

- 发票开具前提条件
  - 客户必须先完善账户信息且进行实名认证后，才能开具发票。
  - 发票申请需基于订单或月度结算，充值未消费部分无法开具发票。
- 查看路径：进入“费用中心> 发票管理 ”页面

用户指南

模型广场

浏览模型

进入 模型广场 页面，可查看平台所有上架的AI模型。页面以卡片形式展示模型，每张卡片包含模型名称、所属分类标签、模型简介、提供厂商及操作按钮。当前平台共计上架37个模型，涵盖计算机视觉、语音、语义、多模态等多个领域。

筛选与搜索模型

在模型广场页面左侧筛选面板中，可以按厂商、算法类别维度筛选模型，也可在页面顶部搜索框输入模型名称进行搜索。右侧模型列表将根据筛选条件实时更新。

查看模型详情

点击目标模型卡片，进入模型详情页，可查看以下内容：

1. 模型介绍：包含使用场景、版本列表、流量限制等
2. API文档：该模型的API接口调用方式和参数说明
3. 功能体验：点击可跳转到体验中心在线体验该模型

体验中心

体验中心可在线体验文本、语音、视觉等各类模型能力

体验中心提供各类AI模型的在线体验功能，无需编写代码即可直观感受模型能力。体验中心分为三大模块：文本模型、语音模型、视觉模型，通过页面顶部Tab页签切换。

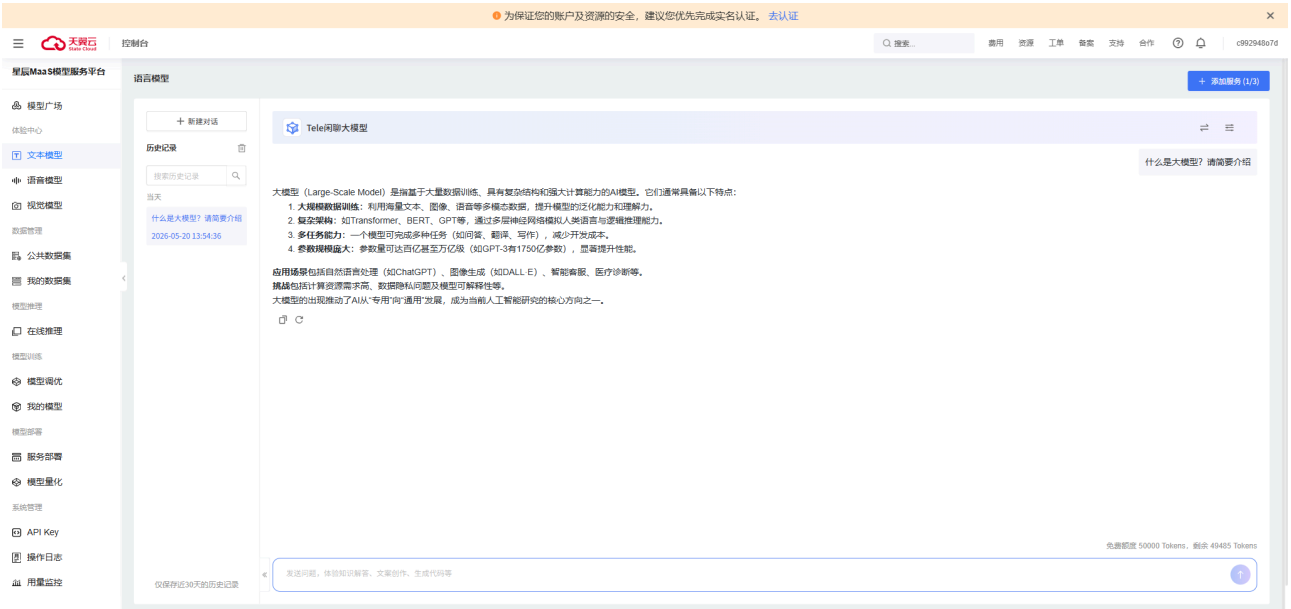
文本模型体验

文本模型体验中心 提供对话式交互界面，支持3个模型，可通过模型名称旁的切换图标切换：

模型名称	模型说明
Tele闲聊大模型	基于TeleChat2-35B基座模型，基于闲聊场景训练的闲聊语义大模型，适用实时对话场景，模型回复偏口语化、简短的风格。
Telechat通用问答大模型	通用问答大模型，具备文本理解、文本创作生成、逻辑推理、语言翻译、数学计算等多种场景的问答能力。
Tele复杂推理大模型	复杂推理模型通过多步骤逻辑推导解决数学、编程等难题，具备长上下文处理能力，高效精准。

发起对话

1. 进入体验中心-文本模型页面，在底部输入框中输入问题或指令。
2. 按下回车键或点击发送按钮，模型将流式返回回答结果。页面底部显示免费额度消耗情况（文本模型免费额度为50000 Tokens）。



调整模型参数

点击模型名称旁的设置图标，可调整以下推理参数：

参数名称	默认值	说明
Temperature	0.3	控制生成文本的多样性。较高的温度值会使生成的文本更加随机和多样化，而较低的温度值会使生成的文本更加确定和一致。
Top_p	1.00	影响输出文本的多样性，取值越大，生成文本的多样性越强。

其他操作

1. 快捷话题：页面提供知识问答、内容创作、工作效率、生活服务、技术支持等预设话题分类，点击即可体验；点击"换一换"可刷新预设话题
2. 新建对话：点击"新建对话"按钮清空当前对话内容，开启新的对话
3. 历史记录：点击右上角"历史记录"展开历史对话列表，支持搜索，仅保存近30天的记录
4. 添加服务：最多可同时添加3个模型服务进行对比体验

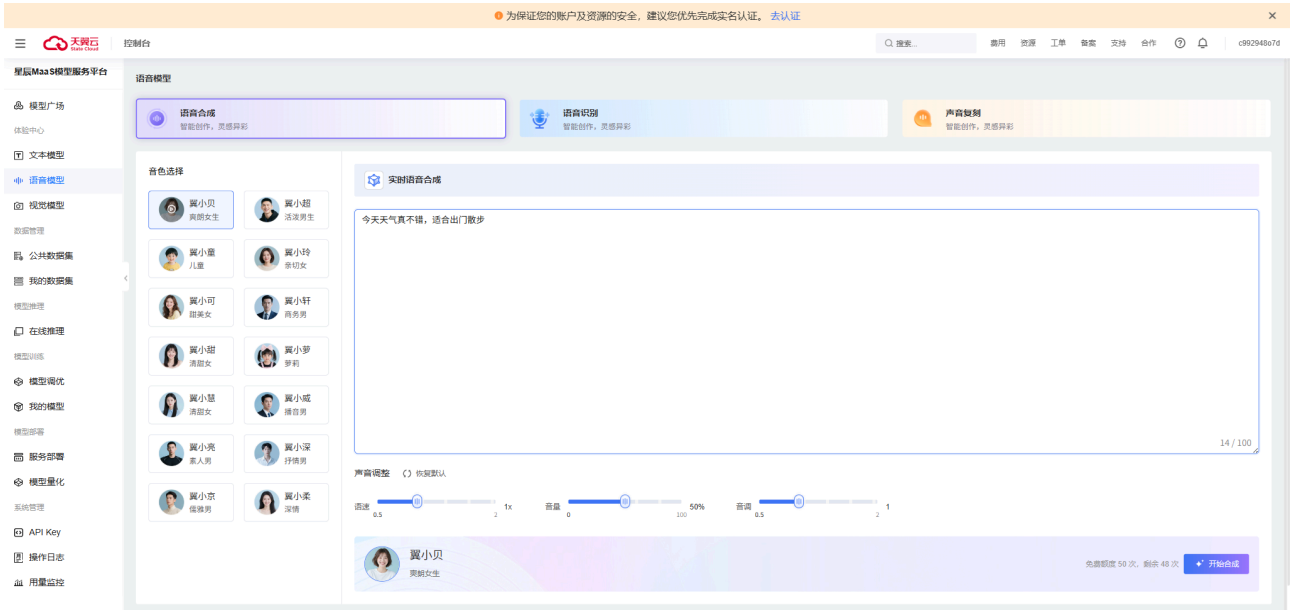
语音模型体验

语音模型体验中心 提供语音合成、语音识别和声音复刻三大功能。

语音合成

1. 语音合成功能（实时语音合成）支持将文本内容转换为语音播报。
2. 在语音模型页面顶部选择"语音合成"Tab页签。
3. 在左侧音色选择区域选择目标发音人（如"翼小贝"）。
4. 在右侧声音调整区域调整语速、音量、音调等参数（可选）。

在文本输入框中输入需要合成的文字内容（不超过100字），点击"开始合成"按钮。



声音调整参数：

参数	默认值	说明
语速	1x	调节语音播放速度，取值范围0.5x ~ 2x
音量	50%	调节语音输出音量，取值范围0 ~ 100
音调	1	调节语音音调高低，取值范围0.5 ~ 2

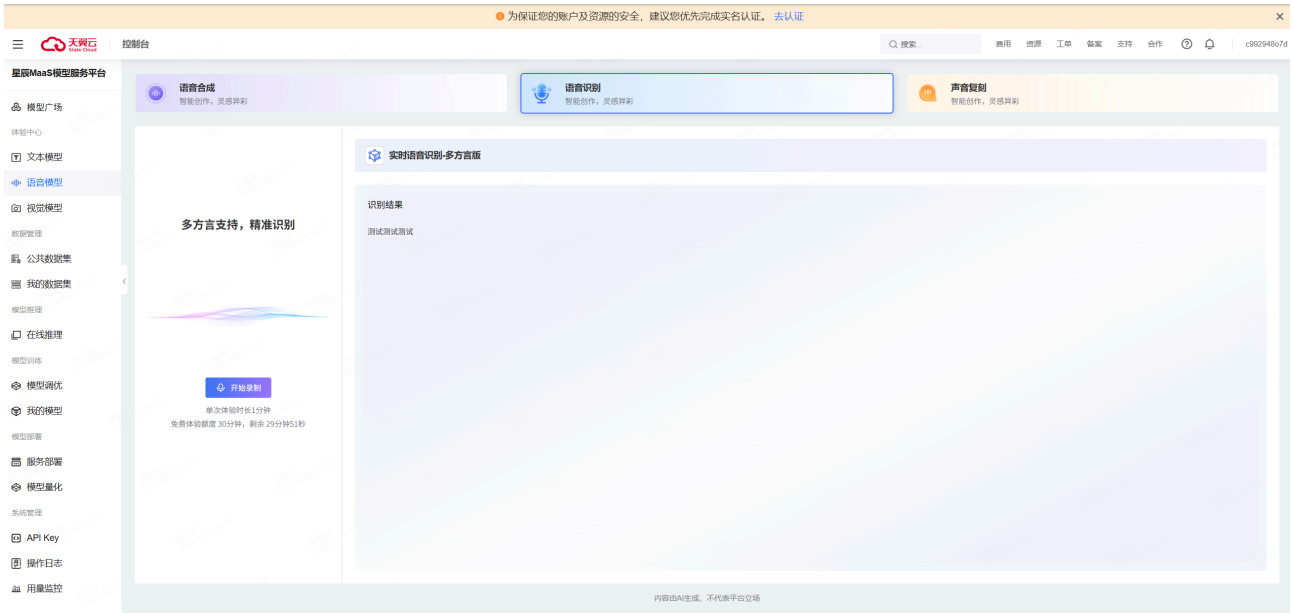
提供"恢复默认"按钮可一键重置所有参数。免费额度为50次。

语音识别

语音识别功能（实时语音识别-多方言版）支持将语音实时转换为文字，业内首家单模型同时支持中文、英文及60+种方言自由混说。

1. 在语音模型页面顶部选择"语音识别"Tab页签。
2. 点击"开始录制"按钮进行语音录制（单次体验时长1分钟）。
3. 录制过程中，识别结果将实时显示在页面右侧区域。
4. 再次点击按钮结束录制。

免费体验额度为30分钟，页面显示剩余时长。



声音复刻

声音复刻功能只需5秒的语音采样，即可精准复刻用户音色、口音、说话风格及背景环境。

1. 在语音模型页面顶部选择"声音复刻"Tab页签。
2. 录制音频：点击"开始录制"，用普通话大声朗读页面显示的参考文案（可点击"换一换"更换文案），录制过程请避免环境噪音。
3. 完成录制后，在下方文本输入框中输入需要合成的内容（不超过100字）。
4. 点击"合成试听"按钮，使用复刻的声音合成语音并试听。

免费额度为20次。

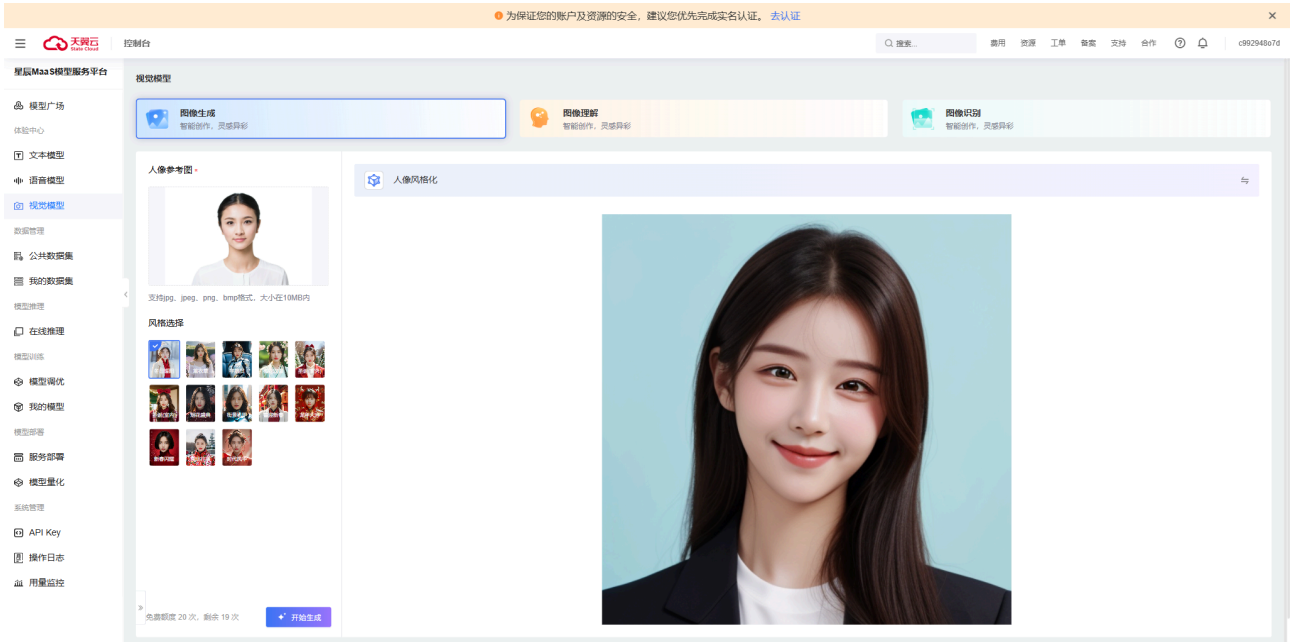
视觉模型体验

视觉模型体验中心 提供图像生成、图像理解和图像识别三大功能。

图像生成（人像风格化）

图像生成功能支持对人像进行风格化处理，上传人像照片后选择目标风格即可生成风格化人像图像。

1. 在视觉模型页面顶部选择"图像生成"Tab页签。
2. 上传人像参考图：支持拖拽或点击上传，支持jpg、jpeg、png、bmp格式，大小在10MB以内。
3. 选择目标风格：平台提供13种预设风格（冬日暖阳、薰衣草、宇航员、梅花汉服、圣诞室外、圣诞室内、烟花盛典、街景机甲、喜迎新春、龙年大吉、新春闪耀、东北往事、时代风华）。
4. 点击"开始生成"按钮，等待生成结果。

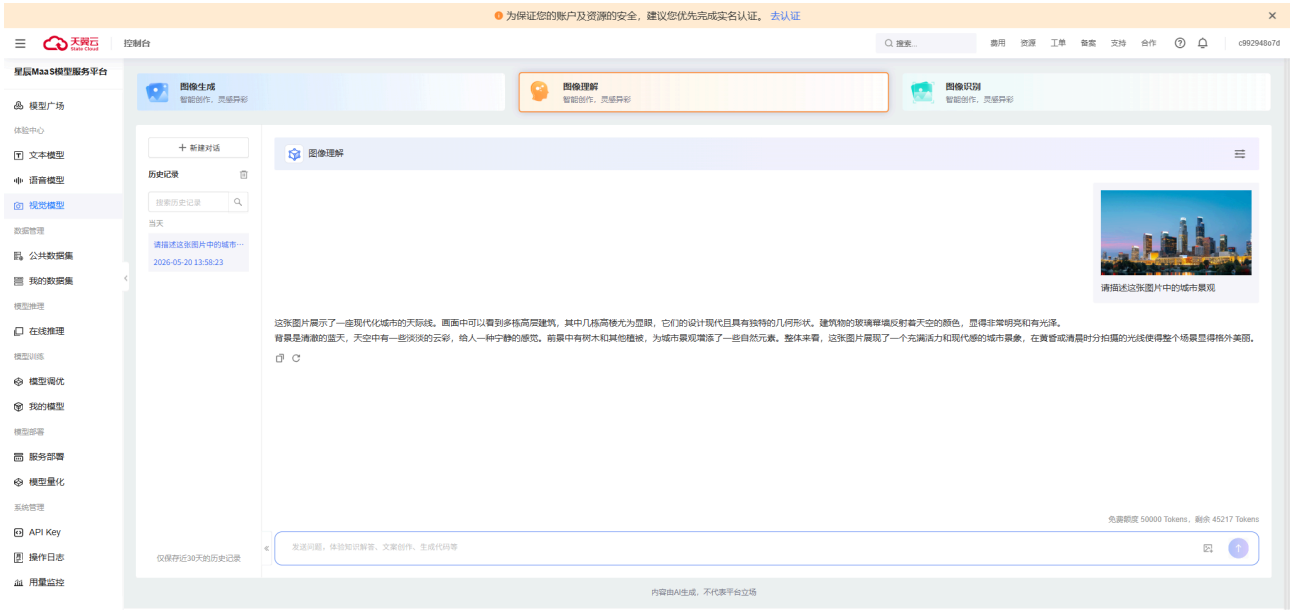


免费额度为20次。

图像理解

图像理解功能基于多模态理解大模型，能够分析理解文本、图像等多种模态数据。

1. 在视觉模型页面顶部选择"图像理解"Tab页签。
2. 通过输入框旁的上传按钮或拍照按钮上传图片。
3. 在输入框中输入关于图片的问题，如"请描述这张图片中的城市景观"，按下回车发送。
4. 模型将分析图片并返回回答结果。



图像理解功能同时支持新建对话、历史记录和搜索功能，仅保存近30天的记录。免费额度为50000 Tokens。

图像识别

图像识别功能提供三大识别能力，通过Tab页签切换：脸人机非全目标解析、明厨亮灶、文字识别。支持上传图片（jpg/jpeg/png/bmp，10MB以内）或输入图片URL进行检测。

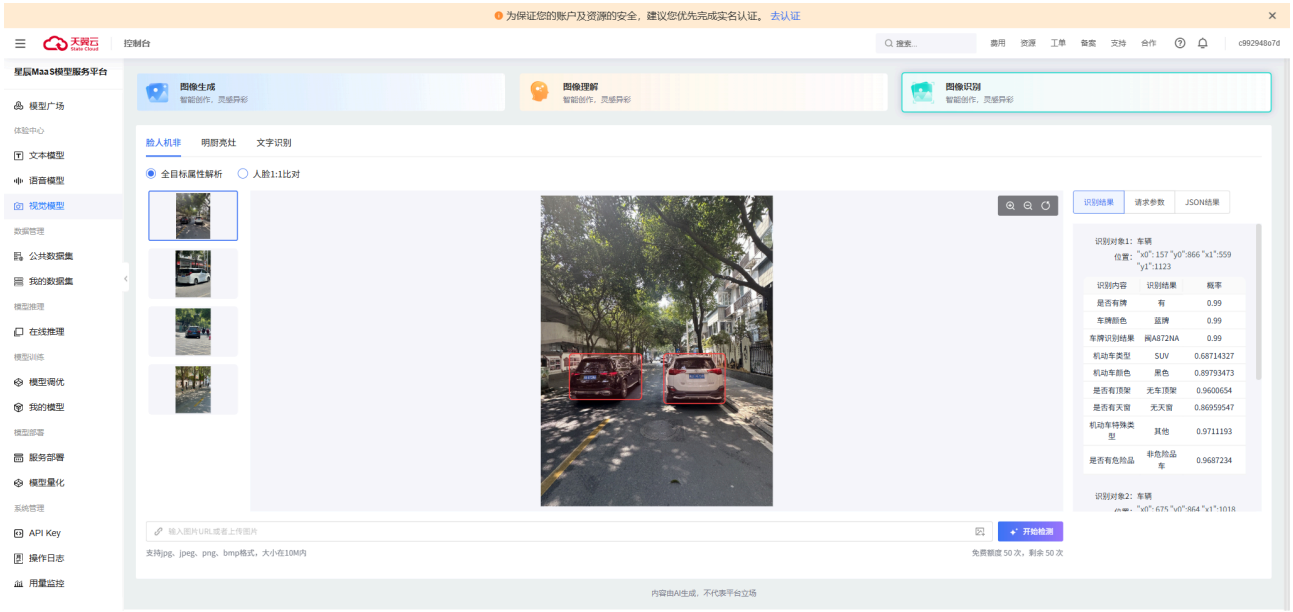
脸人机非全目标解析

支持两种检测模式：

1. 全目标属性解析：对图像中人脸、人体、机动车、非机动车进行识别和解析，输出目标位置坐标及详细属性信息（如车牌号、车牌颜色、机动车类型/颜色、顶架、天窗、危险品标记等），每个属性均包含概率值。
2. 人脸1:1比对：上传两张人脸图片进行相似度比对，支持生活照、证件照、身份证芯片照等多种照片类型。

操作示例——全目标属性解析：

1. 在图像识别页面选择"脸人机非"Tab页签，保持默认"全目标属性解析"模式。
2. 上传图片或点击页面提供的示例图片，然后点击"开始检测"。



检测结果分为三个Tab页签查看：识别结果、请求参数、JSON结果。免费额度为50次。

明厨亮灶

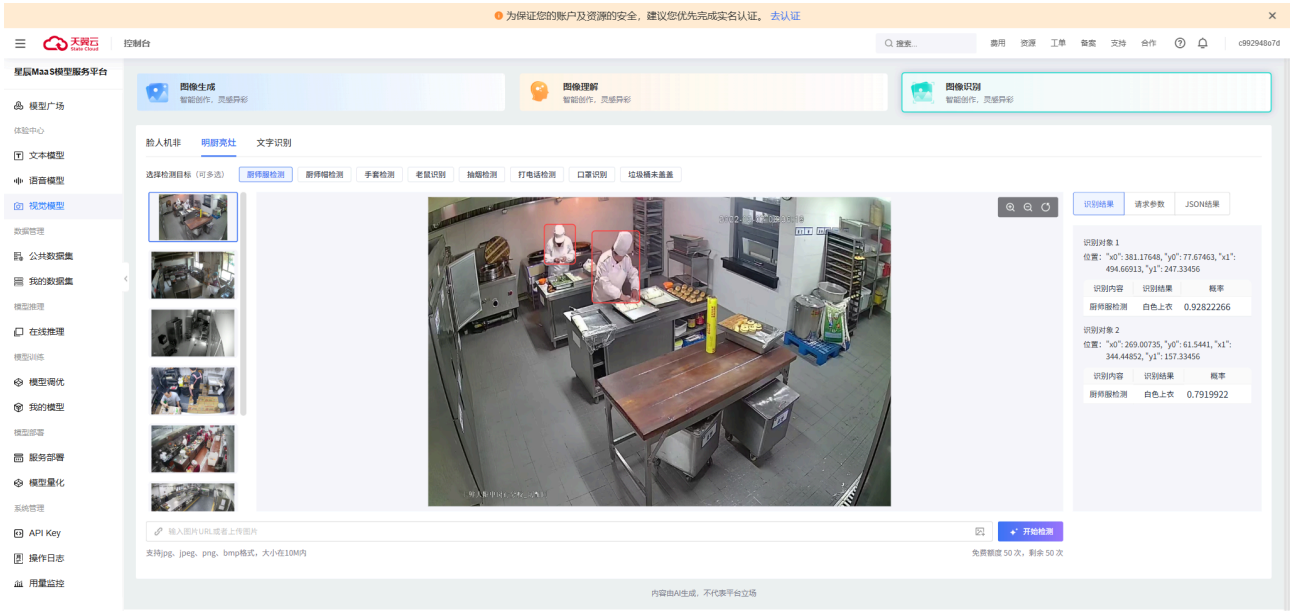
针对厨房场景的多目标检测能力，支持同时选择多个检测目标，包含以下8项：

- 厨师服检测：检测穿着指定颜色上衣或围裙的情况
- 厨师帽检测：实时监测戴厨师帽颜色和未戴情况
- 手套检测：识别佩戴手套情况，支持各种颜色的胶皮/医用手套
- 老鼠识别：针对后厨、仓库等场景提供老鼠检测识别
- 抽烟检测：识别嘴边附近吸烟行为
- 打电话检测：识别贴耳边打电话的行为
- 口罩识别：检测人员配套口罩情况，支持多种口罩类型
- 垃圾桶未盖盖：识别未正常盖盖的垃圾桶

操作示例：

在图像识别页面选择"明厨亮灶"Tab页签，勾选需要检测的目标。

上传厨房场景图片或选择示例图片，点击"开始检测"，查看识别结果。



检测结果以表格形式展示。免费额度为50次。

文字识别

文字识别功能提供三种识别模式：

- 通用文字识别：基于深度学习技术，提供多场景、高精度的整图文字检测和识别服务
- 电子签名识别：基于深度学习，智能检测电子手写签名内容
- 印章识别：支持对合同签章页、委托函等文件上的印章进行智能定位及识别

操作示例——通用文字识别：

1. 在图像识别页面选择"文字识别"Tab页签，保持默认"通用文字识别"模式。
2. 选择示例图片或上传含文字的图片，点击"开始检测"。

检测结果提供三个Tab页签：识别结果、请求参数、JSON结果。免费额度为50次。

数据管理

公共数据集

本文为您展示星辰MaaS模型服务平台-公共数据集模块相关操作。

星辰MaaS模型服务平台包括公共数据集模块，可以实现共享数据集，支持数据集的检索、预览与创建，为用户提供便捷的数据资源，支撑模型训练与调优。

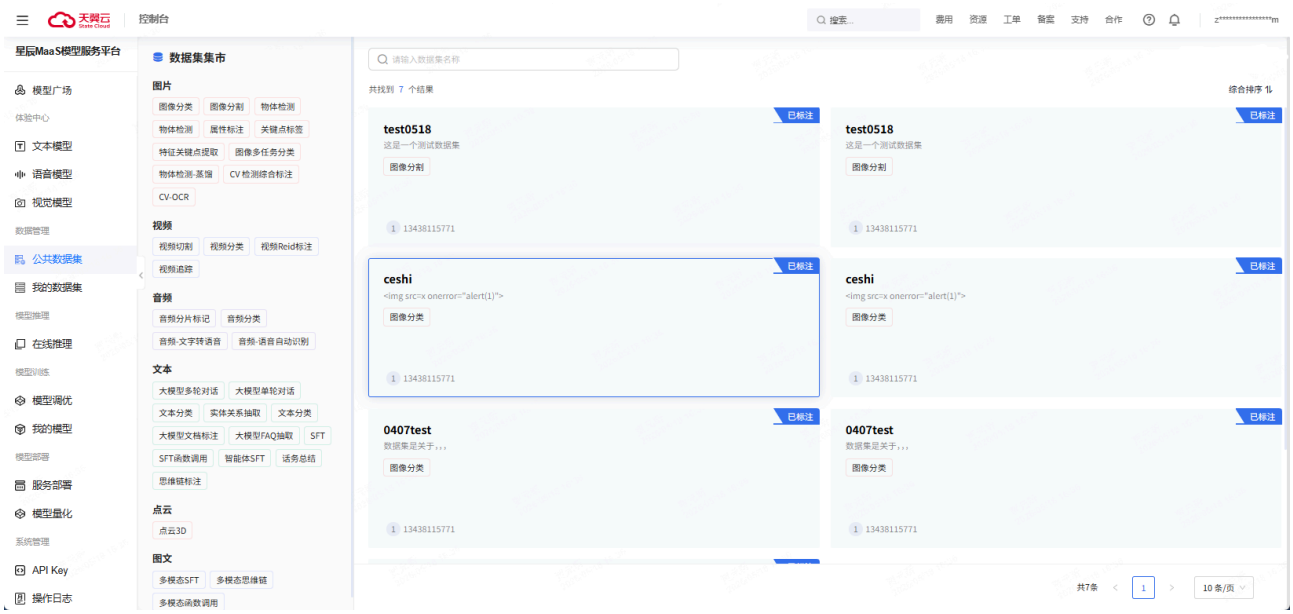
操作步骤

1. 登录 星辰MaaS模型服务平台 。
2. 在左侧导航栏选择“公共数据集”。
3. 点击“创建数据集”，可新增数据集。
4. 在搜索框输入数据集名称，可以查找数据集，或通过数据集集市分类，快速筛选数据集。

操作流程

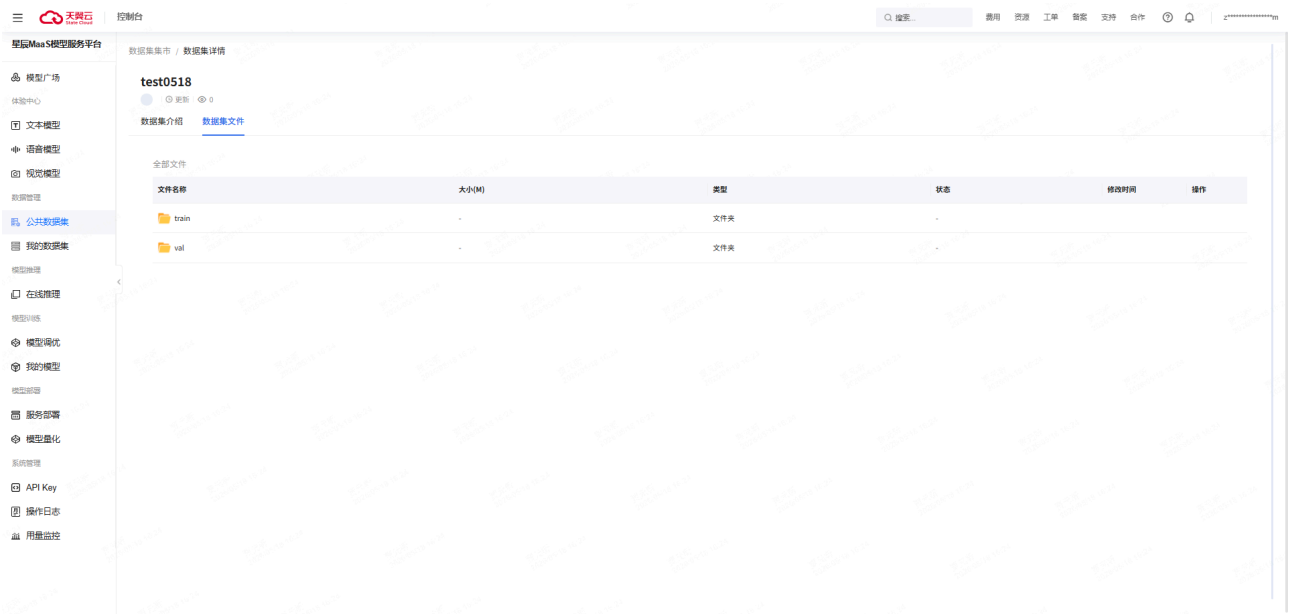
数据集查询

操作说明：可点击数据集的二级分类如：图片分类，也可在输入框输入数据集的名称，列表可筛选出符合条件的数据集。

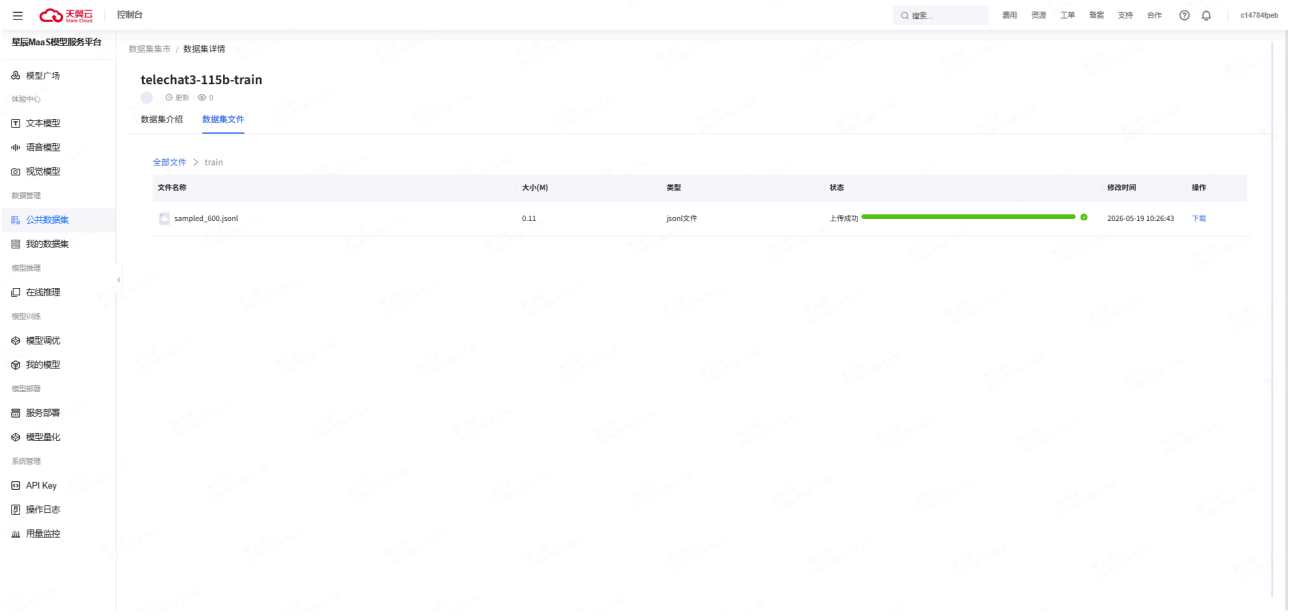


数据集详情

操作说明：点击单个数据集进入，数据集介绍时展示文件的名称、大小、数据集类型、数据集状态和修改时间。



操作说明：点击进入文件夹，可以查看数据集名称、大小、类型、上传状态和修改时间，并支持下载数据集。



我的数据集

本文为您展示星辰MaaS模型服务平台-我的数据集模块相关操作。

星辰MaaS模型服务平台包括我的数据集模块，可以实现私有数据集，支持数据集的检索、预览与创建，为用户提供便捷的数据资源，支撑模型训练与调优。

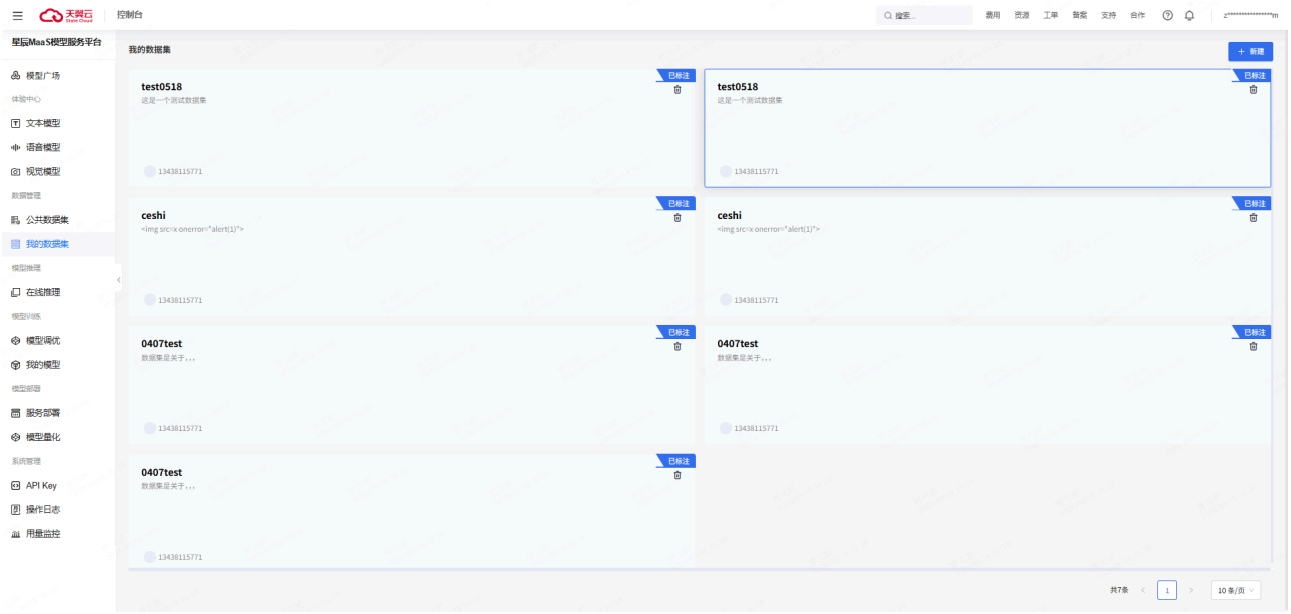
操作步骤

1. 登录 星辰MaaS模型服务平台 。
2. 在左侧导航栏选择“我的数据集”。
3. 点击“新建”，可新增数据集。
4. 可以管理个人数据集，进行删除或者编辑操作。

操作流程

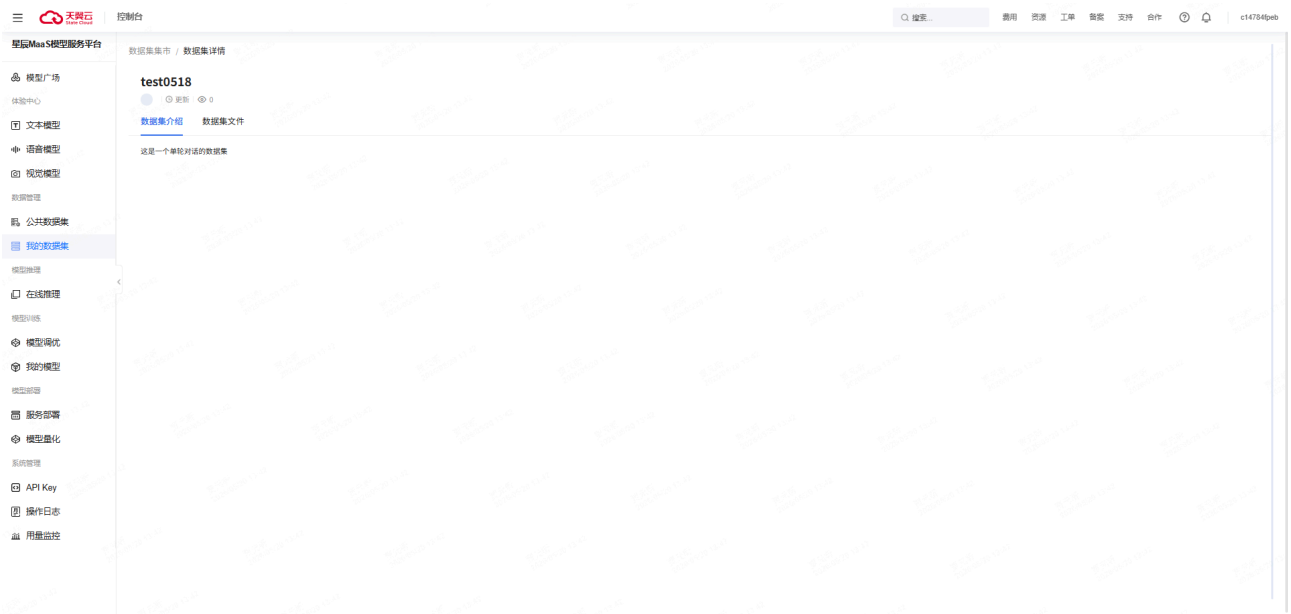
数据集列表

操作说明：查看所有用户自己上传的数据集，支持数据集新建和删除。

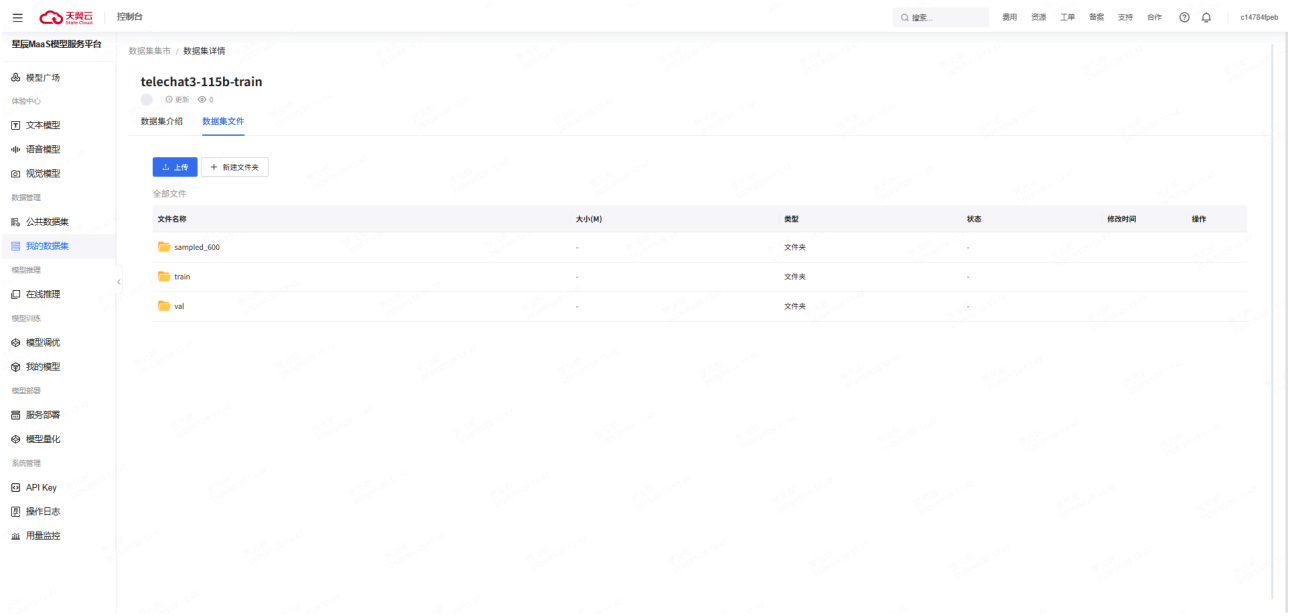


数据集详情

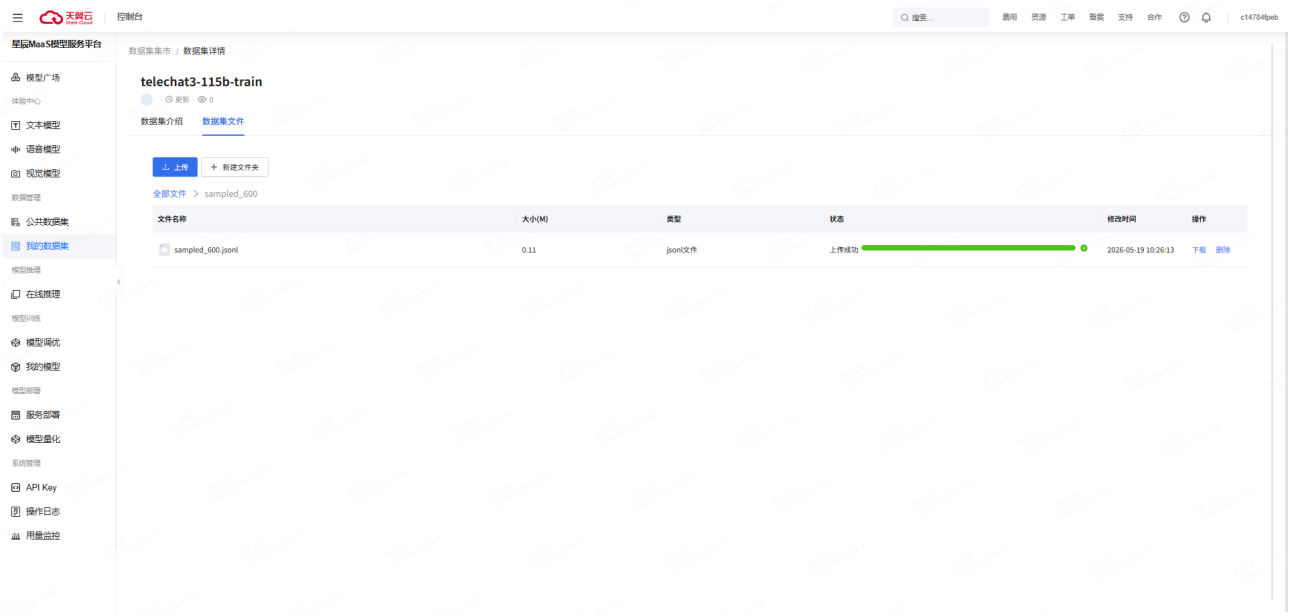
操作说明：点击单个数据集进入，数据集介绍时展示的数据集的描述内容。



操作说明：点击【数据集文件】可查看数据的具体内容。展示文件的名称、大小、数据集类型、数据集状态和修改时间。

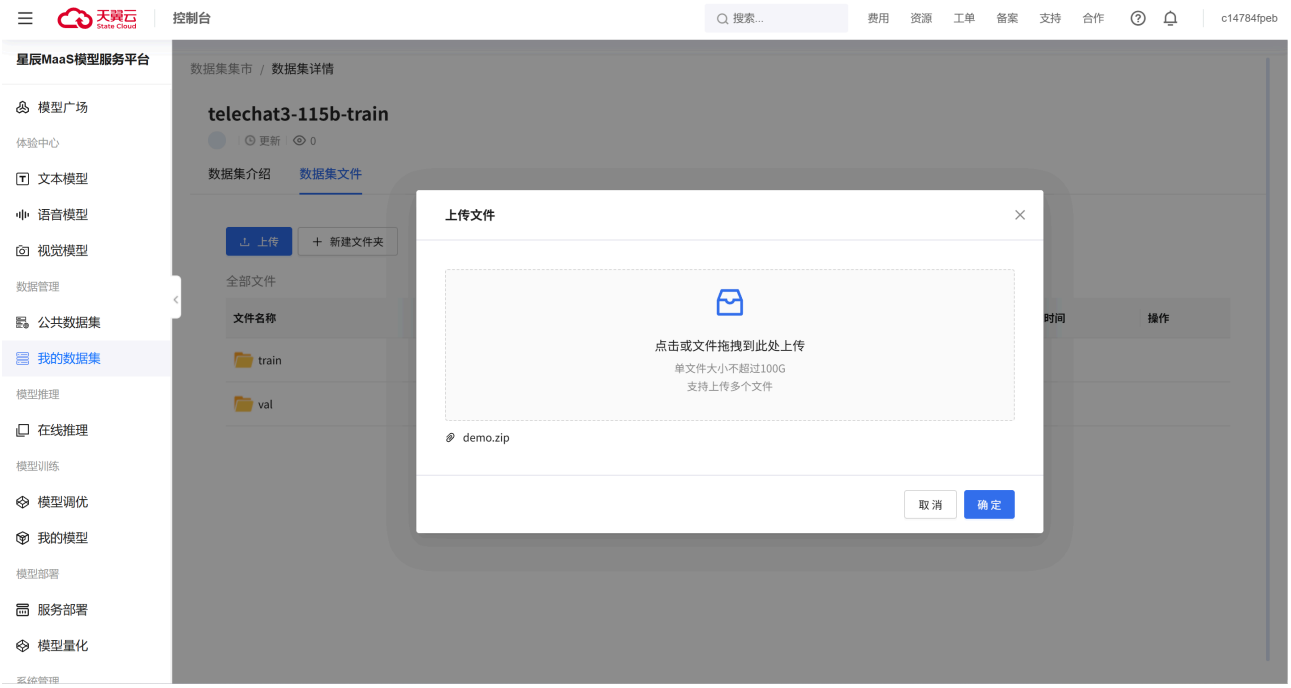


操作说明：点击进入文件夹，可以查看数据集名称、大小、类型、上传状态和修改时间，并支持下载和删除数据集。

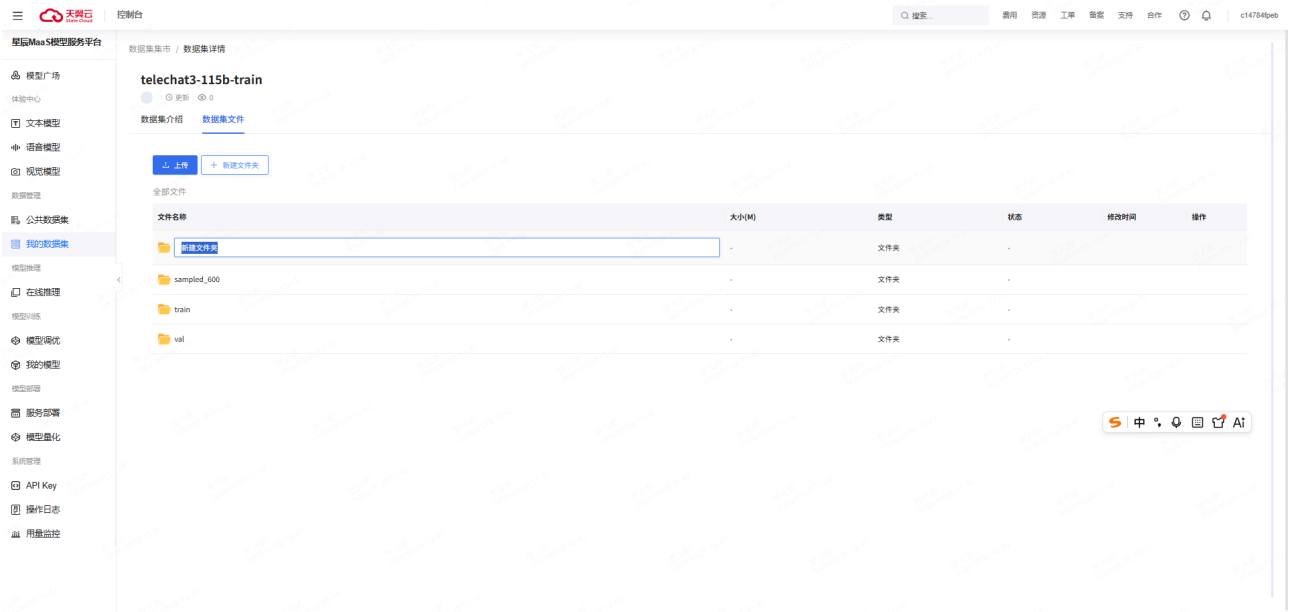


数据集上传

操作说明：在数据集集市点击【上传】进入。可以点击或文件拖拽到此处上传，支持单文件大小不超过100G，支持上传多个文件。需要将数据集转为压缩文件上传。



操作说明：用户点击【新建文件夹】可以创建文件夹。支持修改文件夹名称。



模型推理

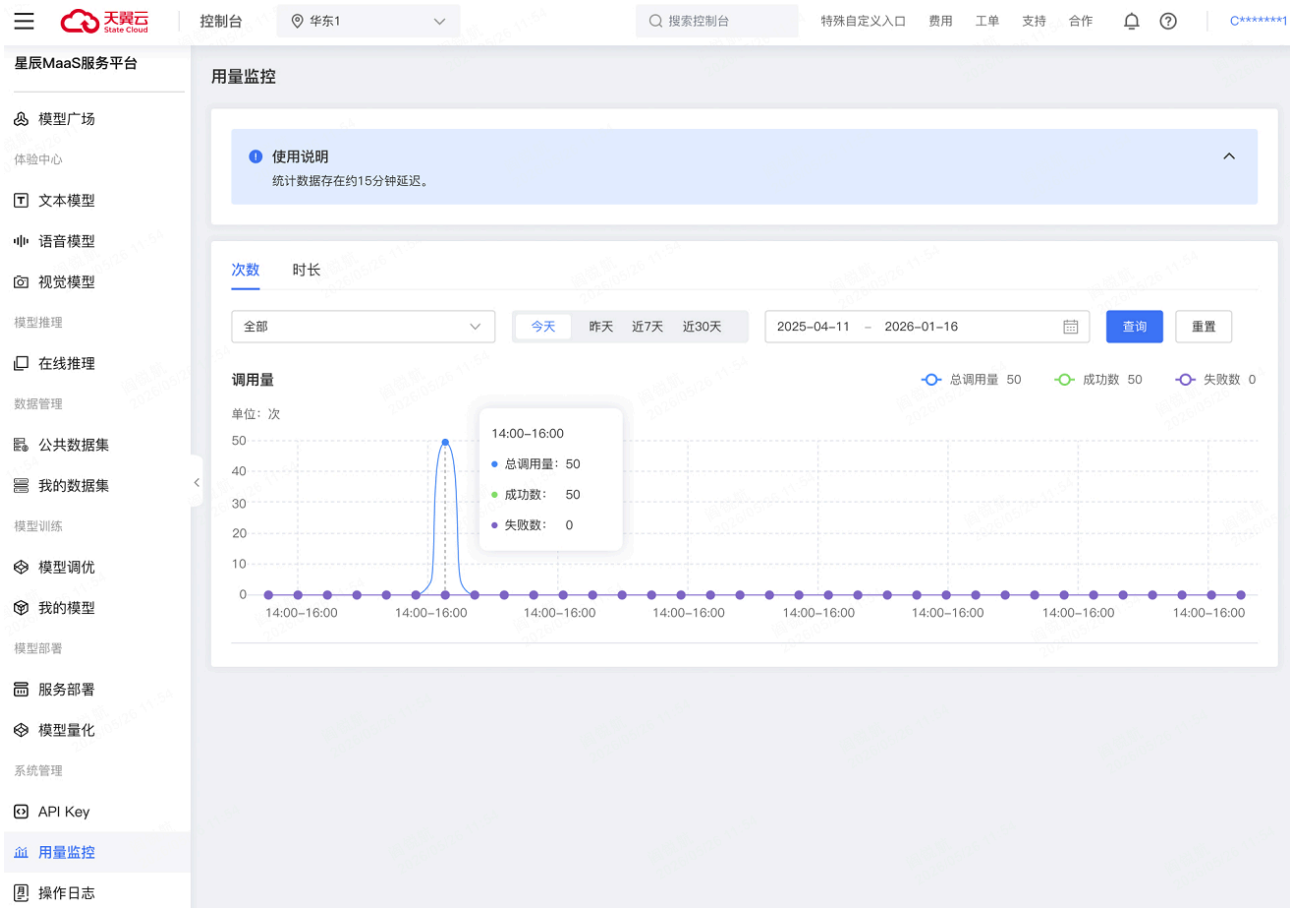
在线推理 模块提供各模型服务的后付费开通和管理功能。开通后付费后即可通过API调用模型推理能力。

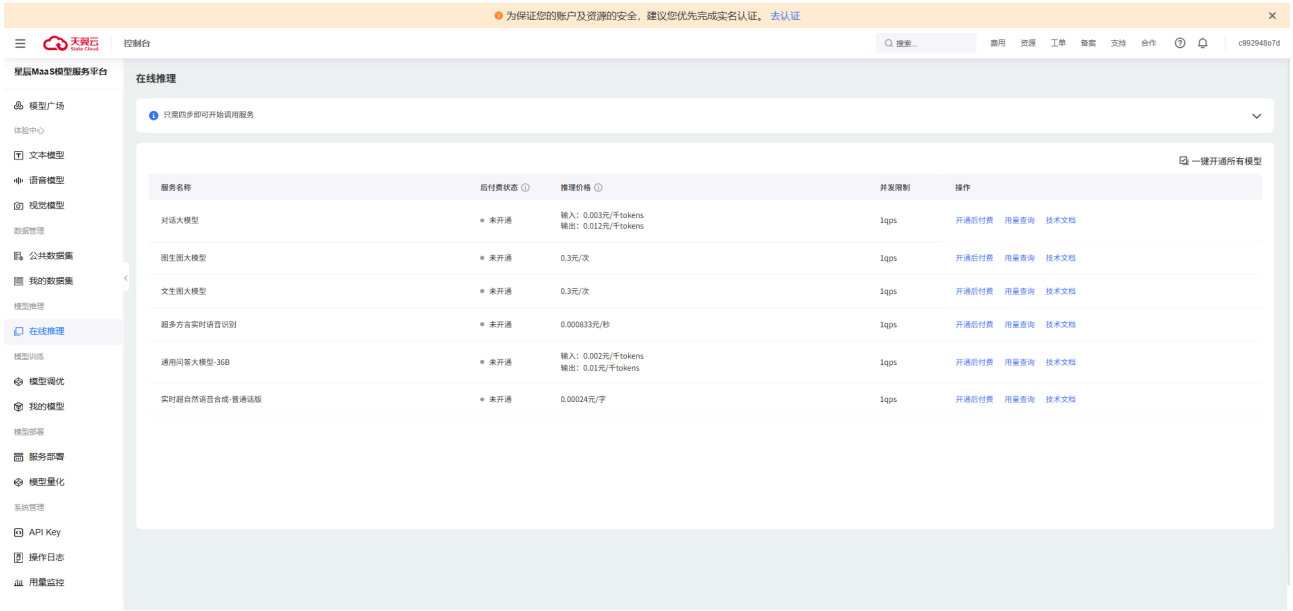
1. 在对应的模型处，点击"开通后付费"按钮，确认开通。
2. 开通后，点击"技术文档"查看API调用接口和参数说明。
3. 使用API Key和接口地址，按照技术文档调用推理服务。

每个推理服务均提供以下操作：

1. 开通后付费：点击“开通后付费”，开通该推理服务的后付费模式，开通后即可通过API调用

2. 用量查询：点击“用量查询”，查看该推理服务的调用用量统计
3. 技术文档：点击“技术文档”，查看该推理服务的API接口调用文档





模型训练

模型调优

本文为您展示星辰MaaS模型服务平台-模型调优模块相关操作。

模型调优提供零门槛调优功能，用户无需编写代码，仅需选择或上传数据集，即可自动完成模型微调，快速定制出贴合特定场景的专业模型。

操作步骤

1. 登录 星辰MaaS模型服务平台 。
2. 在左侧导航栏选择“模型调优”。
3. 点击“新建”，可新增调优任务，包括任务名称、基础模型、基础模型版本、数据集等。
4. 创建成功后启动模型调优任务，调优成功后，可以发布成为新模型。同时支持模型调优任务删除和编辑等管理功能。
5. 通过关键词搜索可以快速查询到模型调优任务。

操作流程

步骤一：调优任务列表

操作说明：点击菜单【模型训练-模型调优】进入页面。列表页展示团队内创建的所有调优任务，点击【新建】进行任务创建。

步骤二：调优配置

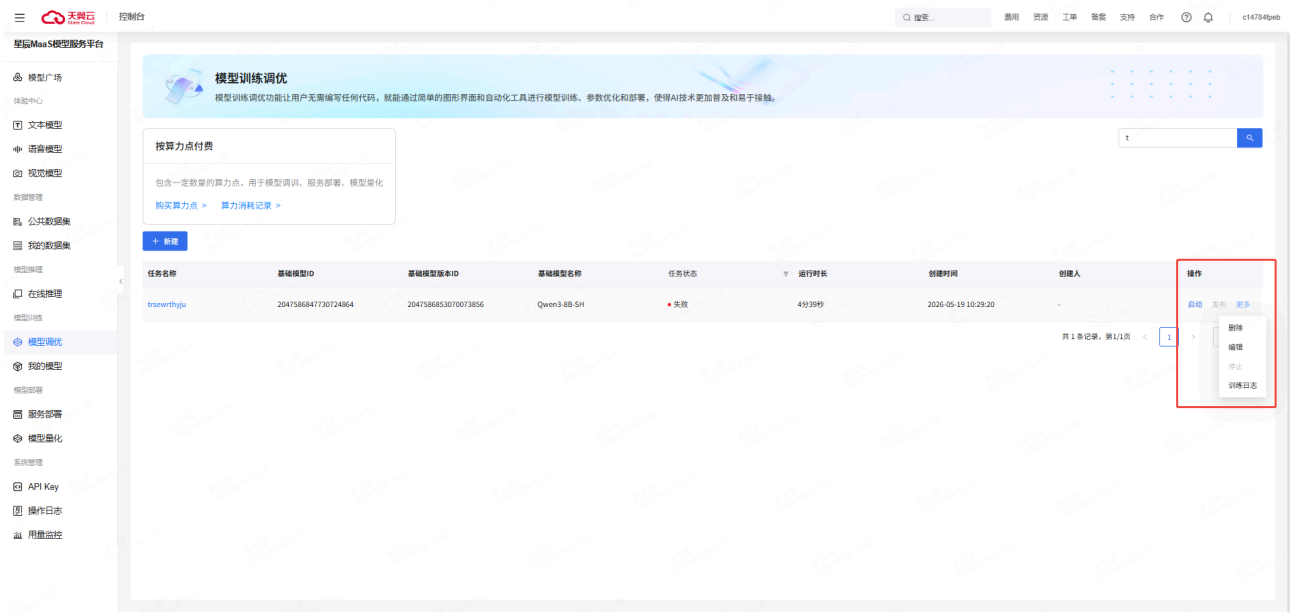
操作说明：

1. 创建模型调优任务名称和备注信息；

2. 选择需要调优的模型；
3. 选择是否使用模型训练默认配置，指训练资源的配置，如果选择否，就会展开资源配置的列表；
4. 选择训练数据集和验证数据集；

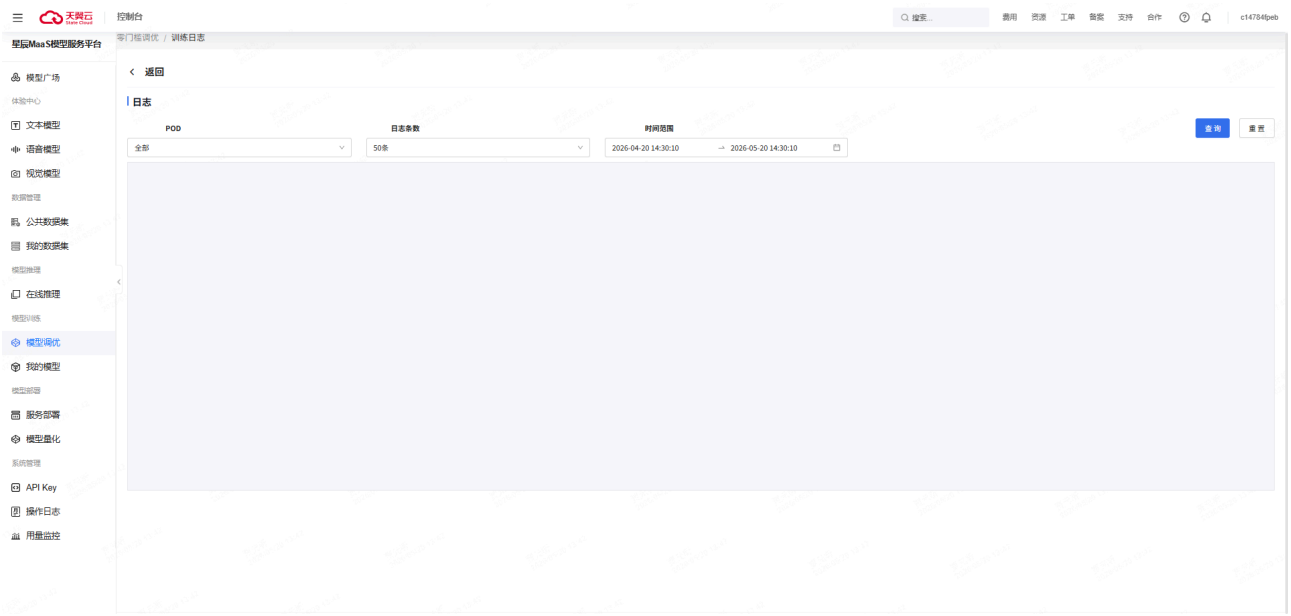
步骤三：调优任务管理

操作说明：在模型调优列表页面，可在搜索框直接搜索任务的名称，点击键盘的【enter】键或者点击查询按钮，列表展示搜索结果。搜索到创建的任务后点击【启动】任务便会运行起来。可以很方便的里看到每条任务所处的状态。当需要暂停任务时，直接点击【停止】按钮即可。想删除任务时，直接点击【删除】按钮即可。支持编辑任务，点击【编辑】按钮即可。



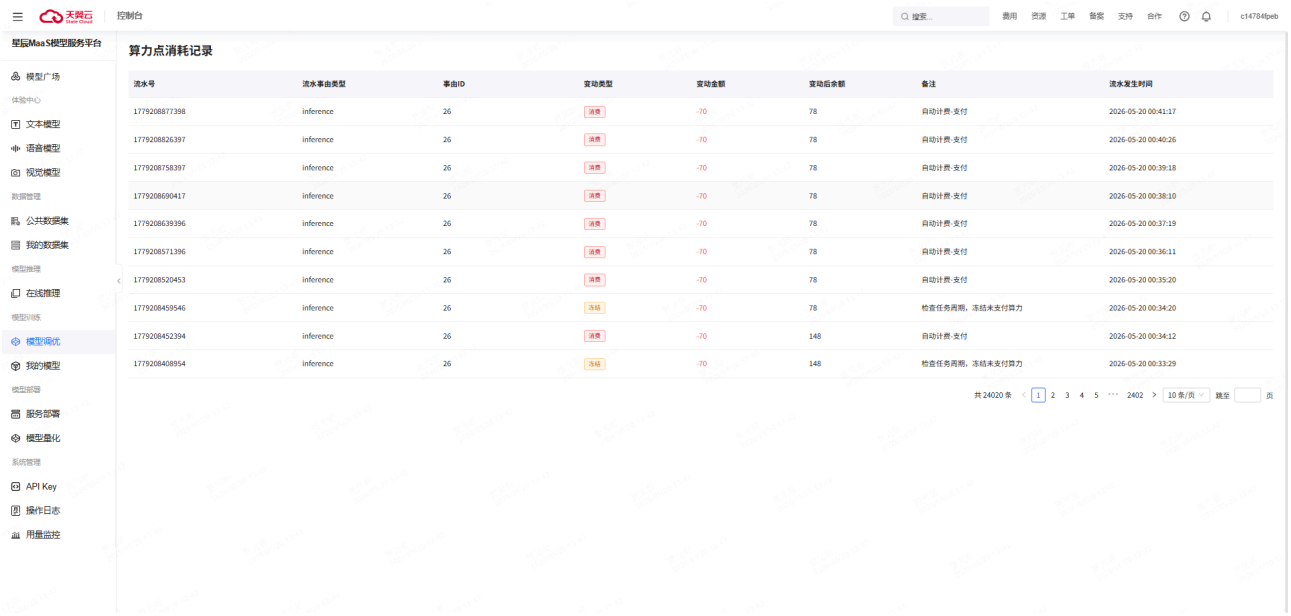
步骤四：查看训练日志

操作说明：在模型训练时模型训练列表页面操作区域可点击【训练日志】，可以实现查看训练相关的日志指标。支持按POD，日志条数，时间范围进行查询，同时支持一键【重置】查询条件。



步骤五：查看算力消耗

操作说明：在模型调优列表页面，可在【按算力点付费】模块，点击【算力消耗记录】查看算力消耗详情，包括流水号、流水事由类型、事由ID、变动类型、变动金额、变动后余额、备注和流水发生时间。当算力不足是，支持用户在线进行充值。



我的模型

本文为您展示星辰MaaS模型服务平台-我的模型模块相关操作。

我的模型模块提供用户个人模型管理，用户可以通过本地化上传模型文件，创建自己的专属模型，并对模型进行管理。

操作步骤

1. 登录 星辰MaaS模型服务平台 。

2. 在左侧导航栏选择“我的模型”。
3. 用户可以对模型进行增删改等管理操作。

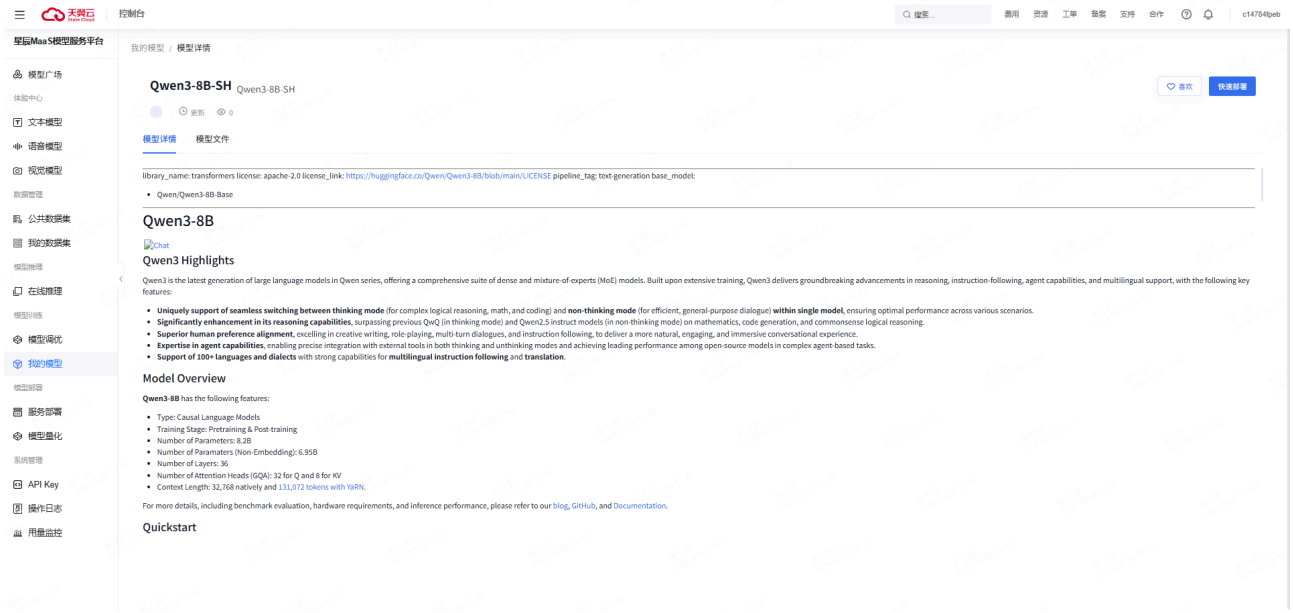
操作流程

模型列表

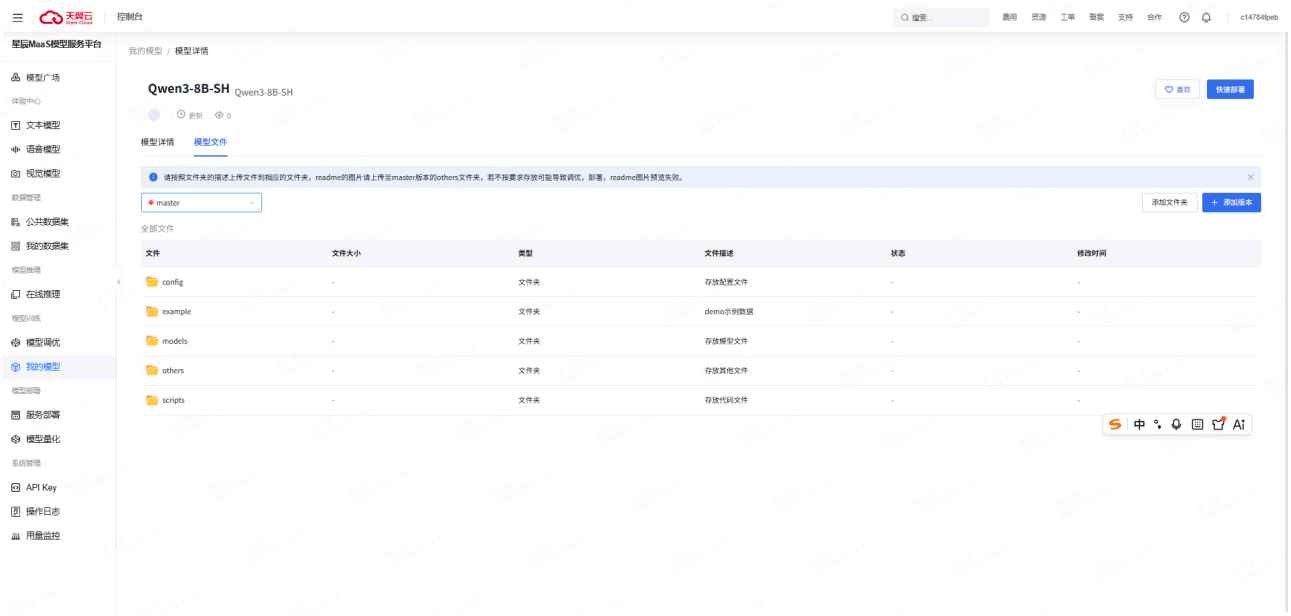
操作说明：点击【模型训练-我的模型】进入我的模型列表，用户可以点击模型进行模型详情查看，可以创建模型。

模型详情

操作说明：在我的模型列表页面，点击【Qwen3-8B-SH】进入，查看模型详情，可以看到模型的详细信息。

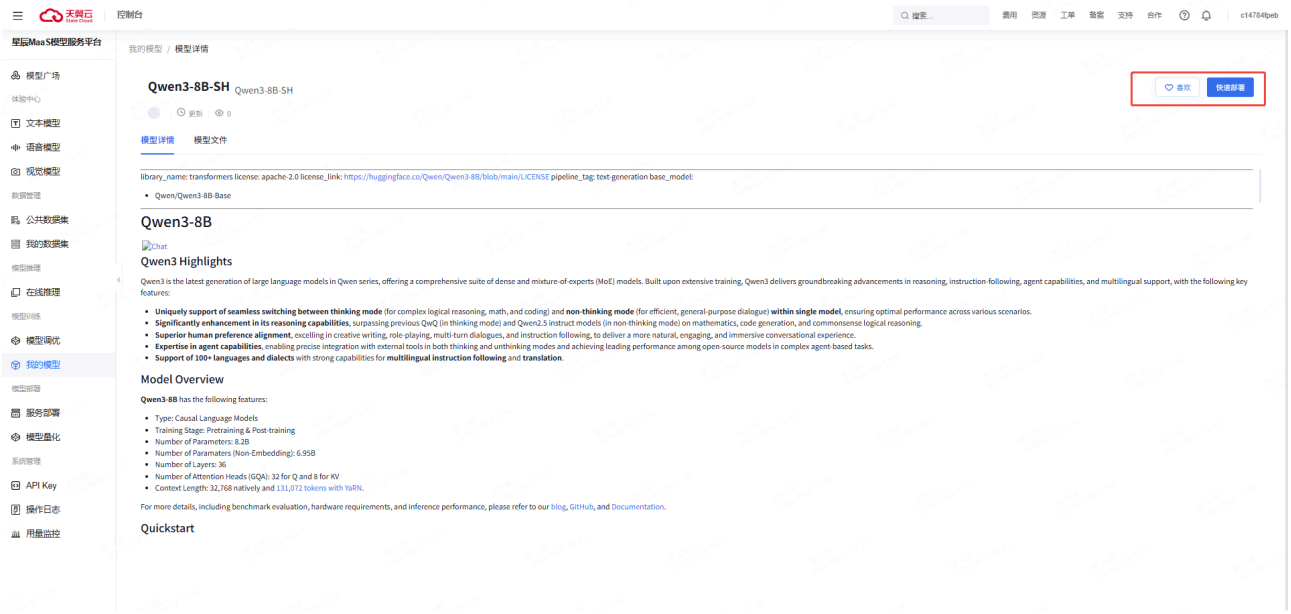


操作说明：在模型详情页面，点击【模型文件】进入，查看模型文件详情，可以看到全部文件，支持添加文件夹，支持添加版本。



操作说明：在模型详情页面，点击【喜欢】可以收藏该模型。

操作说明：在模型详情页面，点击【快速部署】可以跳转到服务部署模块



模型部署

服务部署

本文为您展示星辰MaaS模型服务平台-服务部署模块相关操作。

支持模型一键部署为在线推理服务，提供服务地址供外部调用，并支持服务上下线和管理等功能。

操作步骤

1. 登录 星辰MaaS模型服务平台 。

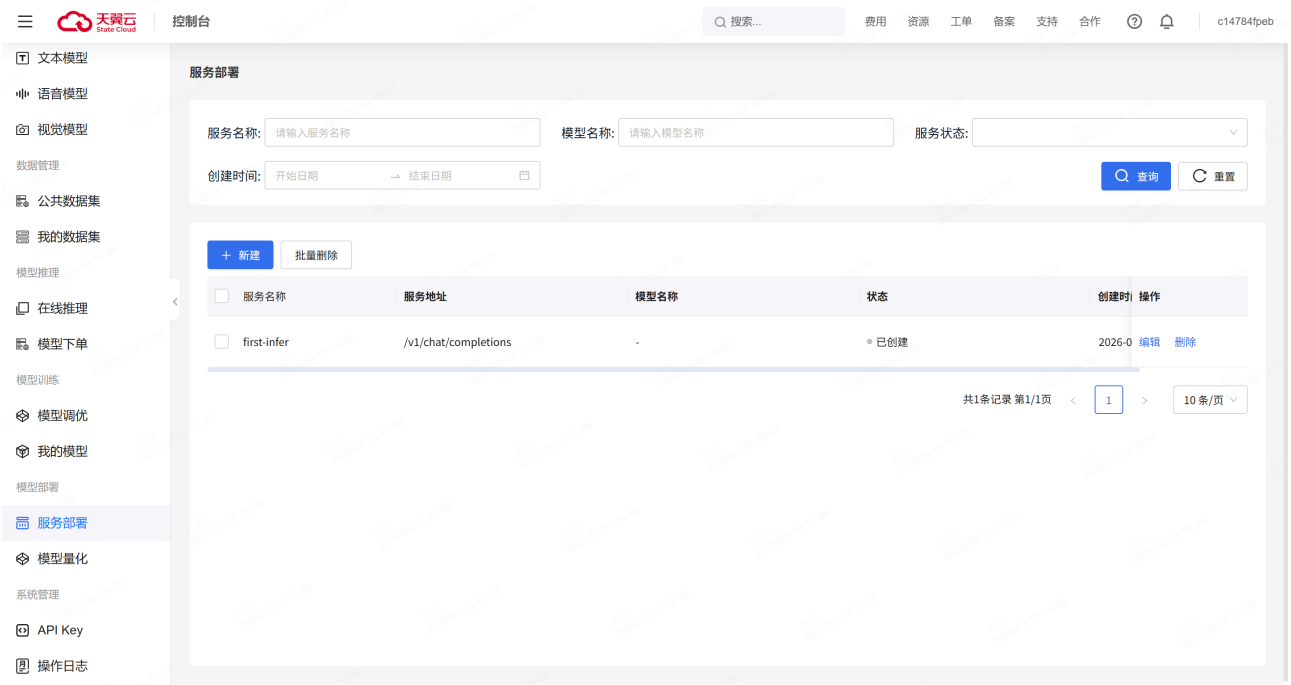
2. 在左侧导航栏选择“服务部署”。
3. 点击“新建”，可创建待部署的模型，部署成功后，系统会生成服务名称及服务地址。
4. 创建成功后可以对模型进行编辑和删除操作。

操作流程

创建服务

服务列表

操作说明：点击【模型部署-服务部署】进入，在《服务列表》用户可输入服务名称，模型名称，服务状态，创建时间的任一值进行模型的查询。

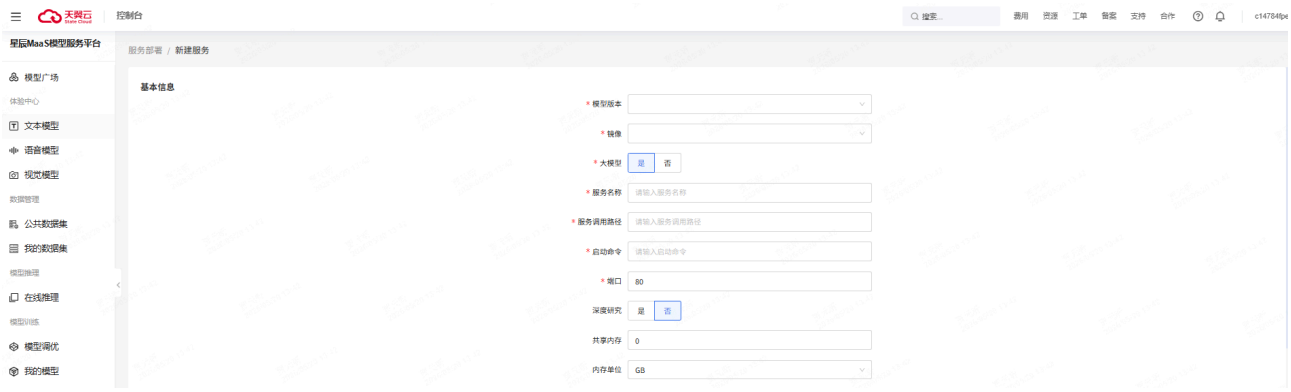


新建服务

操作说明：在《服务列表》点击【新建】，也可在《模型详情页面》点击【快速部署】进入。新建服务包括四个部分信息，服务基本信息、环境变量、算力规格和健康探针配置，分别对服务进行填写，实现模型服务的创建。

- 基本信息

1. 选择模型版本信息；
2. 选择镜像信息；
3. 选择是否为大模型；
4. 填写服务名称信息；
5. 填写服务调用路径信息；
6. 填写启动命令信息；
7. 填写端口信息；
8. 填写共享内存、内存单位信息。



- 环境变量信息

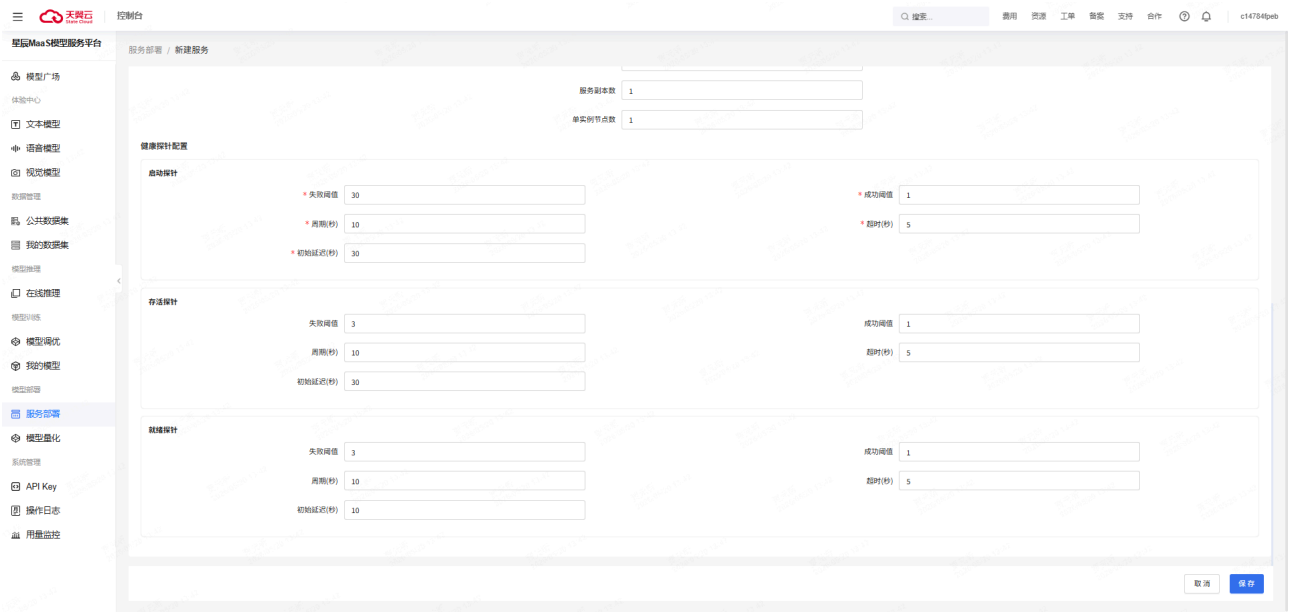
支持在新建模型推理服务页面配置环境变量，包括变量名、变量值等。

- 算力规格信息

支持在新建模型推理服务页面配置算力规格，包括规格类型、服务副本数、单实例节点数。

- 健康探针配置

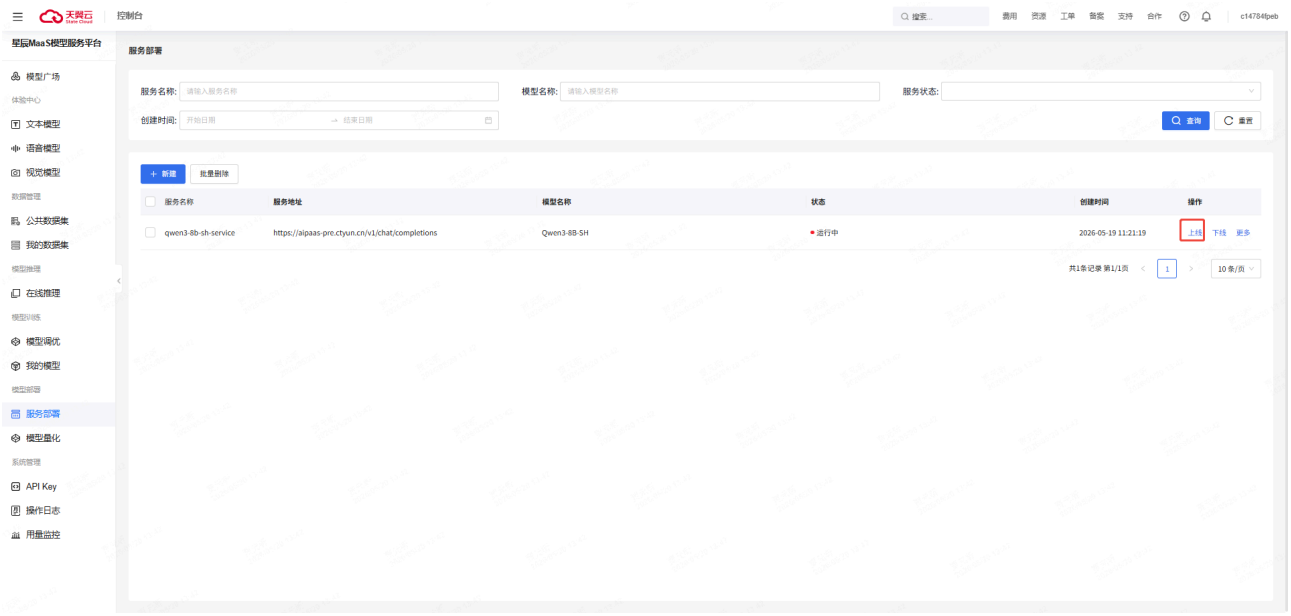
支持在新建模型推理服务页面配置健康探针。包括启动探针、存活探针、就绪探针各自的失败阈值、周期、初始延迟、成功阈值、超时时间等。



服务部署

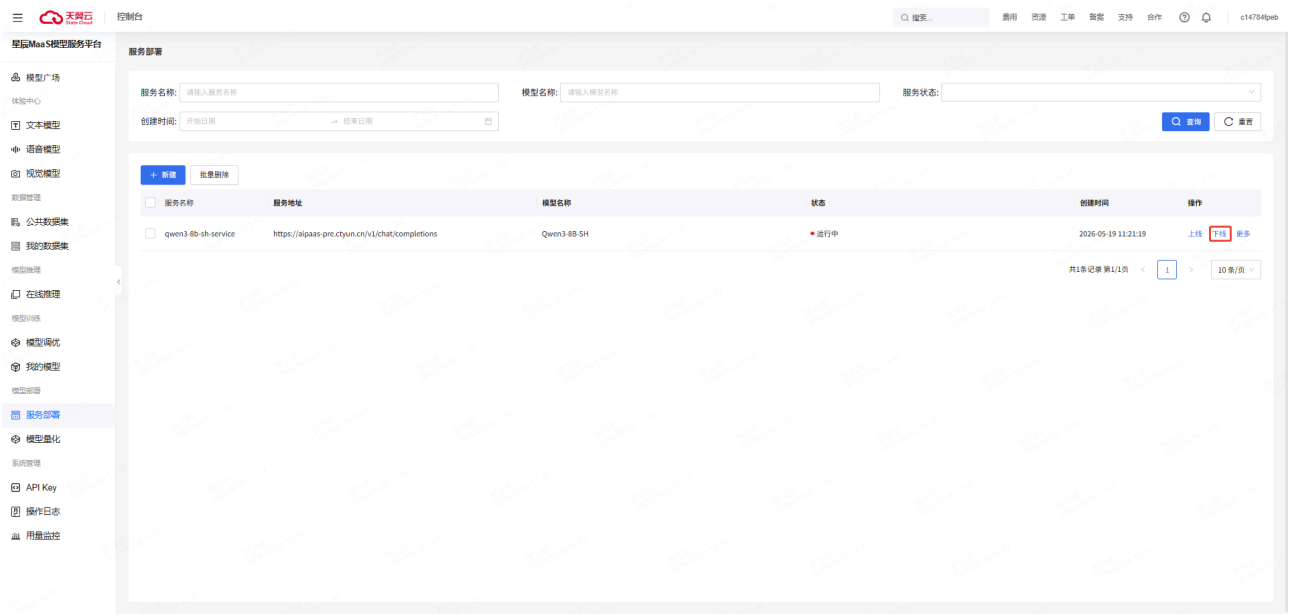
上线

操作说明：任务创建完成后会返回《服务列表》，点击【上线】系统会执行上线流程，部署完成就是已上线的状态。



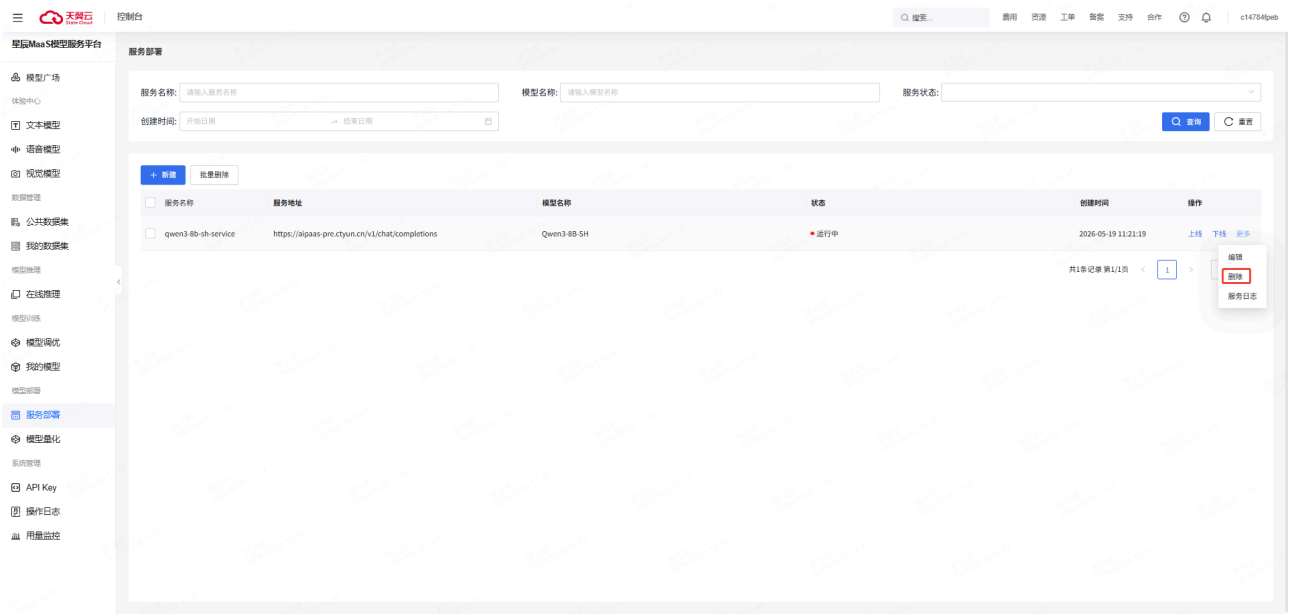
下线

操作说明：上线后的服务想下线则点击【下线】系统会执行下线流程，执行完成就是已下线的状态。



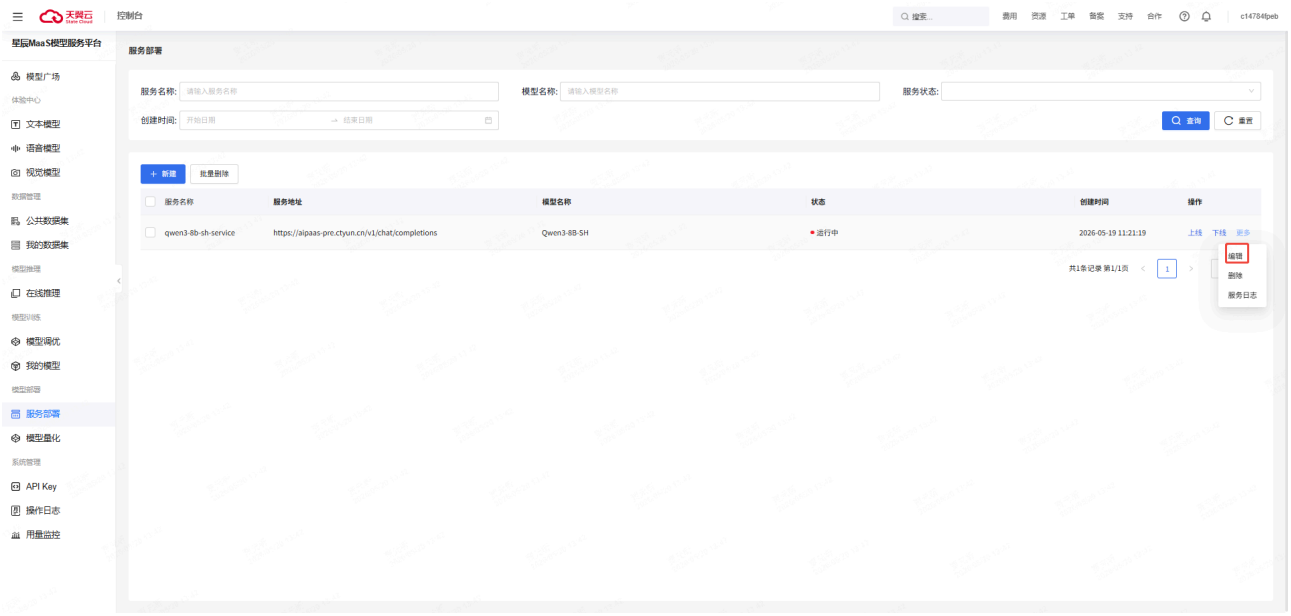
删除

操作说明：在《服务列表》想删除点击【删除】系统会执行删除操作，执行完成就是不显示在模型列表的状态。



编辑

操作说明：在《服务列表》想编辑点击【编辑】系统会执行编辑操作，打开编辑页面，执行完成后保存返回模型列表。



服务日志

操作说明：在《服务列表》点击【服务日志】进入。支持按POD，日志条数，时间范围进行查询，同时支持一键【重置】查询条件。

模型量化

本文为您展示星辰MaaS模型服务平台-模型量化模块相关操作。

模型量化是一种模型压缩技术，其核心思想是将深度学习模型中的权重和激活值从高精度数值（如32位浮点数，FP32）转换为低精度数值（如8位整数，INT8）。

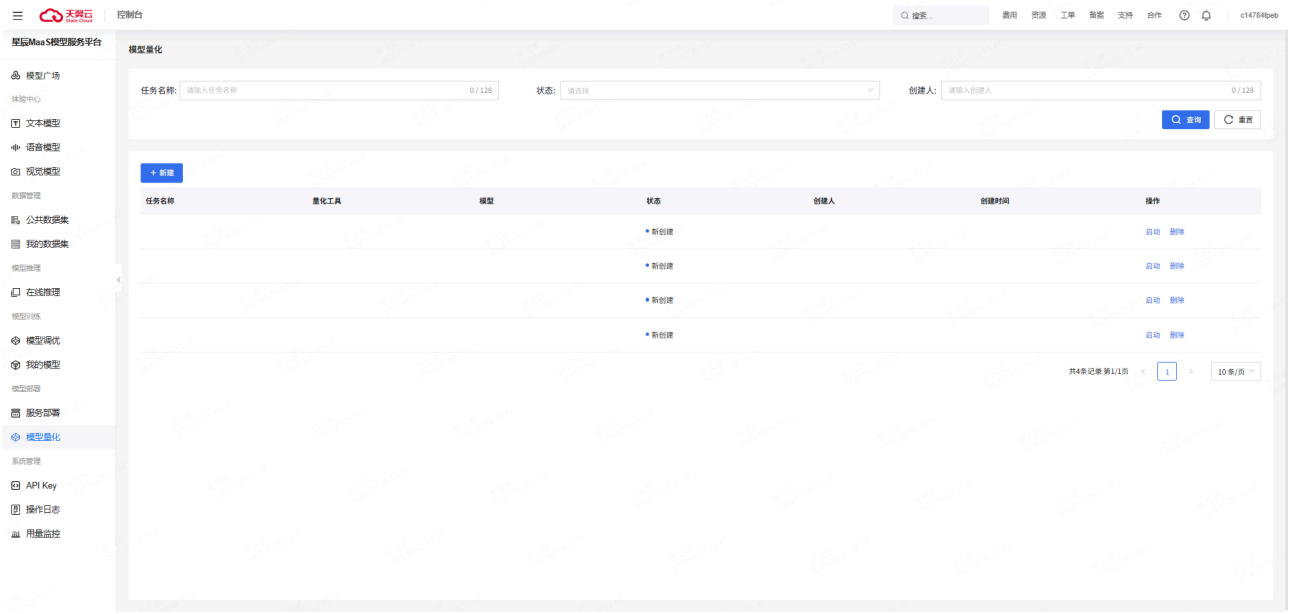
操作步骤

1. 登录 星辰MaaS模型服务平台 。
2. 在左侧导航栏选择“模型量化”。
3. 点击“新建”，可创建待量化的模型，选择量化工具和模型，开始进行模型量化。
4. 创建成功后可以对量化的模型进行编辑和删除操作。

操作流程

步骤一：模型量化列表

操作说明：点击菜单【模型部署-模型量化】进入页面。列表页展示团队内创建的所有量化任务，支持按任务名称、状态、创建人进行任务查询，点击【新建】进行任务创建。用户可以启动模型量化任务，或者删除创建的模型量化任务。



步骤二：新建任务

操作说明：在模型部署-模型量化页面，点击【新建】。进入模型量化功能，点击列表上方的新建按钮，进入新建流程。

1. 填写任务名称、备注信息；
2. 选择需要量化的模型；
3. 选择是否使用模型训练默认配置，指训练资源的配置，如果选择否，就会展开资源配置的列表；
4. 选择量化需要使用的数据集；
5. 选择量化工具：GPTQ、AWQ、W8A8；

系统管理

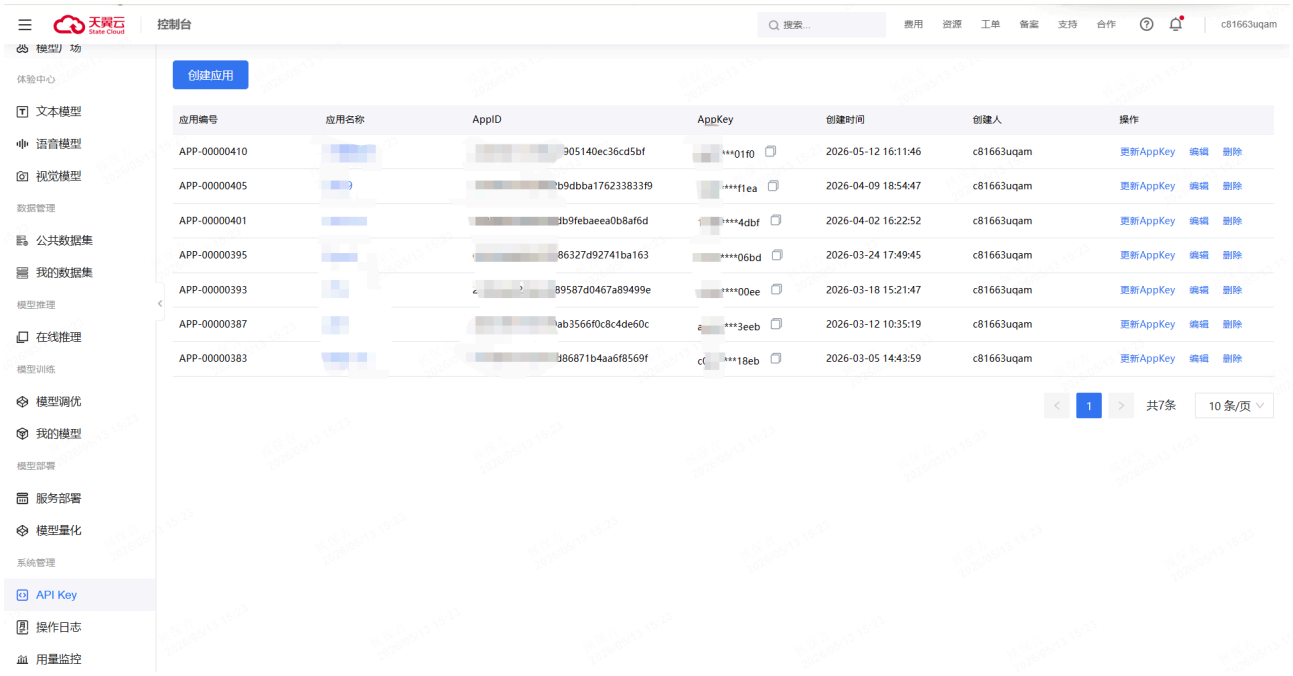
API Key

功能介绍

用户调用模型服务需要获取API调用相关鉴权信息，含创建、编辑、删除应用及获取【AppID】（应用的唯一标识符，用于在平台调用已购买的API接口）、【AppKey】（平台应用的密钥，与 AppID 配合使用，用于生成 API 请求的认证签名（Signature），以确保请求的合法性和安全性）等鉴权信息，功能具体使用说明如下：

创建应用及获取鉴权信息

进入 星辰MaaS模型服务平台 ，访问【系统管理】>【API Key】点击"创建应用”按钮，按步骤完成应用创建，系统自动生成鉴权信息（AppKey、AppID ）



操作日志

本文为您展示星辰MaaS模型服务平台-操作日志模块相关操作。

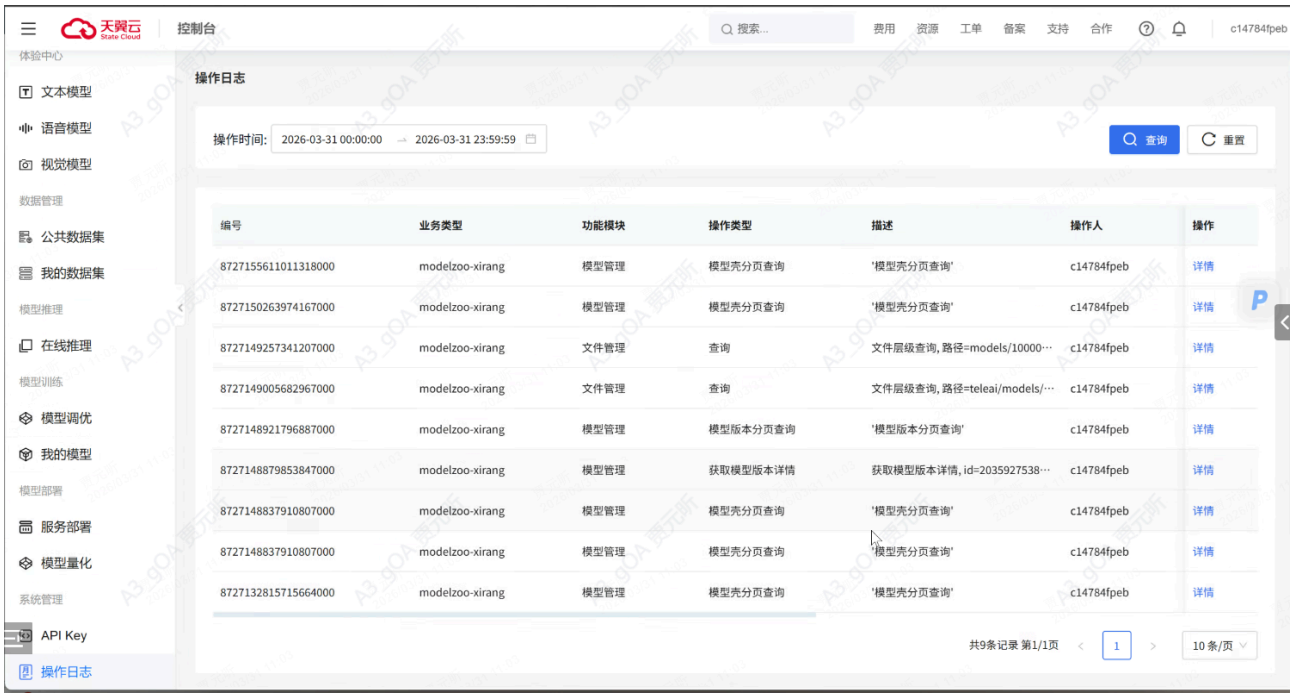
操作日志可以查看用户操作的记录，帮助用户进行审计。

操作步骤

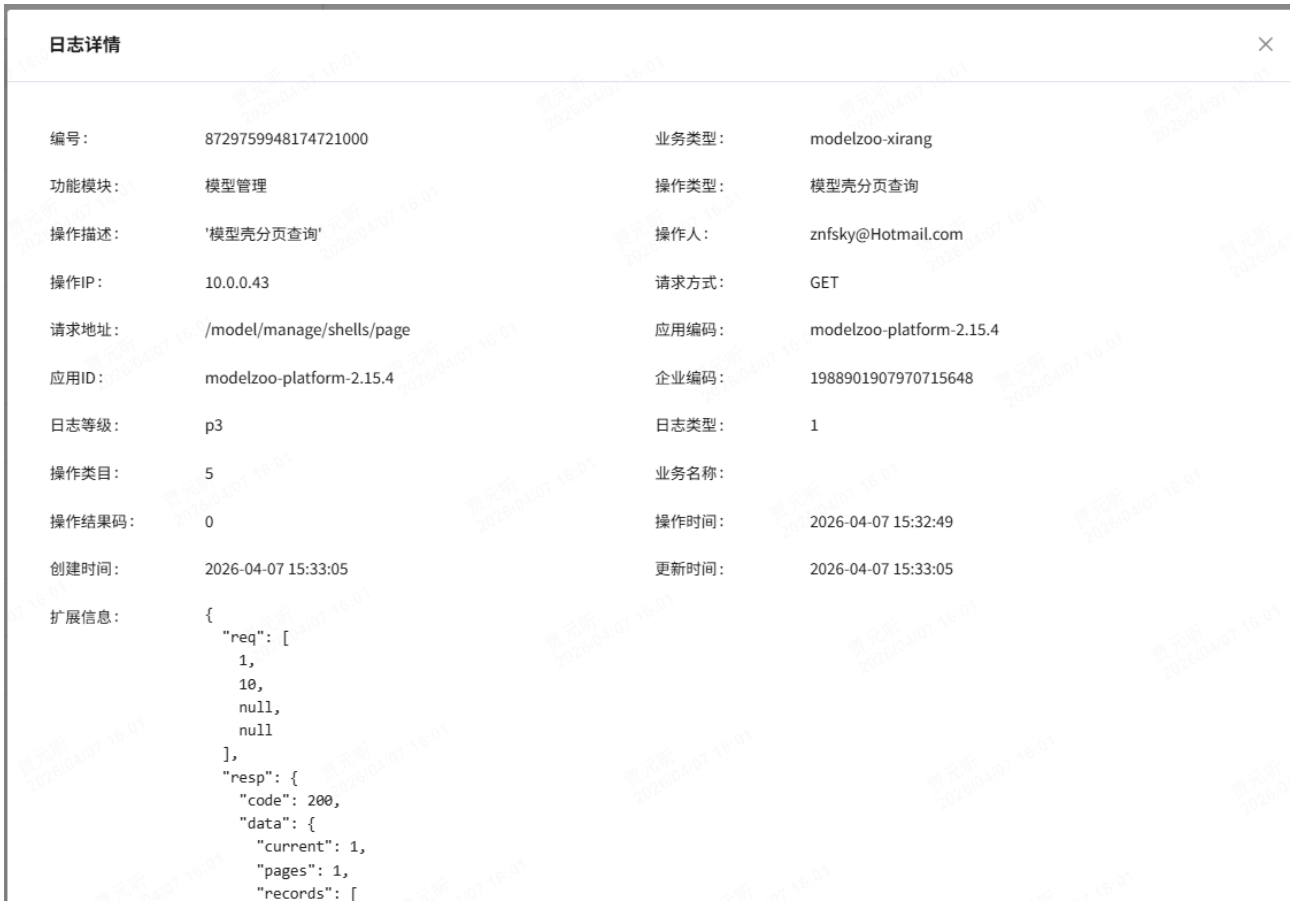
1. 登录 星辰MaaS模型服务平台 。
2. 在左侧导航栏选择“操作日志”。
3. 点击“查询”，可按操作时间筛选操作日志。

操作流程

操作说明：点击【系统管理-操作日志】进入。用户可以查看日志编号、业务类型、功能模块、操作类型、描述、操作人、请求方式、请求地址、操作结果、日志等级、操作IP、操作时间等信息，并支持查看日志详情。支持通过操作时间范围（开始日期→结束日期）筛选日志。



操作说明：在操作日志列表页面，点击【详情】进入。用户可查看日志详细信息，包括展示编号、业务类型、功能模块、操作类型、操作描述、操作人、操作IP、请求方式、请求地址、应用编码、应用ID、企业编码、日志等级、日志类型、操作类目、业务名称、操作结果码、操作时间、创建时间、更新时间、扩展信息及操作请求信息等。



快速入门

准备工作

本文介绍星辰MaaS模型服务平台使用前的准备工作。

注册天翼云账号

在开通和使用星辰MaaS模型服务平台之前，您需要先注册天翼云账号。

1. 注册账号，请参考账号中心- 注册天翼云账号-账号中心-操作指南 - 天翼云
2. 如需实名认证，请参考账号中心- 中国电信天翼云-管理中心

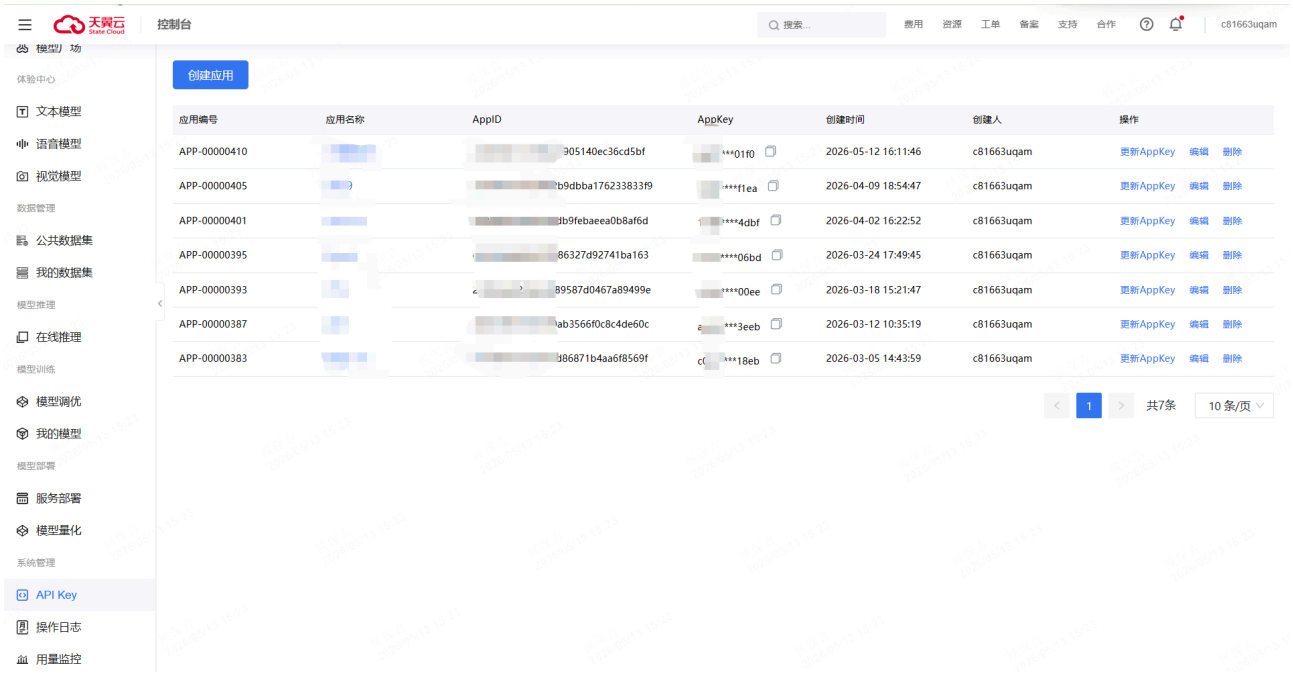
充值主账户

1. 使用星辰MaaS模型服务平台之前，请保证您的账户有充足的余额
2. 请前往 中国电信天翼云-管理中心 为账户充值

快速入门

获取鉴权信息

登录 星辰MaaS模型服务平台控制台 ，访问：系统管理--API Key，创建应用获取AppID 和AppKey


控制台
搜索...
费用
资源
工单
备案
支持
合作
c81663uqam
体验中心
文本模型
语音模型
视觉模型
数据管理
公共数据集
我的数据集
模型推理
在线推理
模型训练
模型调优
我的模型
模型部署
服务部署
模型量化
系统管理
API Key
操作日志
用量监控
创建应用
应用编号
应用名称
AppID
AppKey
创建时间
创建人
操作
APP-00000410
2026-05-12 16:11:46
APP-00000405
2026-04-09 18:54:47
APP-00000401
2026-04-02 16:22:52
APP-00000395
2026-03-24 17:49:45
APP-00000393
2026-03-18 15:21:47
APP-00000387
2026-03-12 10:35:19
APP-00000383
2026-03-05 14:43:59
更新AppKey
编辑
删除
共7条
10 条/页

常见问题

常见问题

本文向您介绍星辰MaaS模型服务平台相关常见问题。

计费类

调优和推理服务支持哪些计费方式?

调优和推理服务支持按算力点计费模式。

调优和推理服务免费算力额度使用规则是什么?

每个账号提供一定额度且限期2周的免费调用量，限期计算以首次使用时间为起始时间，主/子账号共用免费额度及有效期周期额度。

免费额度用完或到期后可开通付费服务继续付费使用，开通付费服务后仍优先消耗剩余免费额度，免费额度用完或到期后自动转入付费使用。

主子账号在算力点额度使用中是什么规则?

免费额度：每个模型被赋予不等的免费额度和免费试用期限，具体免费额度可在页面中查看，免费期限从第一次使用该模型开始计算，免费额度用完或到期后，可以付费开通服务，暂不支持开通付费服务的模型可选择部署为我的服务后继续使用。

- 共享机制：主/子账号共用免费额度及有效期周期额度
- 不可分配：不支持主账号向子账号分配额度
- 有效期触发：主账号任一账号首次使用模型服务即启动2周有效期倒计时
- 独立开通：各账号可单独开通付费服务，操作互不影响。
- 状态独立：各账号页面显示自身付费状态，不关联其他账号。
- 统一扣费：所有子账号消费均从主账号余额扣除。

  举例：子账号A已开通付费，成功后即可付费使用算力点；子账号B未开通，则子账号B不可付费使用算力点。

在线推理服务支持哪些计费方式?

对话大模型、通用问答大模型-36B按调用Token数计费，图生图大模型、文生图大模型按调用次数计费，超多方言实时语音识别按语音时长计费，实时超自然语音合成-普通话版按字数计费。

账户欠费后如何充值?

请前往天翼云官网进行充值，详细操作参见 在线充值-费用中心-资金管理-余额充值 - 天翼云 。

为什么账户欠费后仍在出账?

欠费自账单出具起算，而账单出账通常存在 约 1 小时的延迟。在此期间，系统按照实际使用量正常出账。

操作类

平台已预置的模型有哪些?

进入在线推理模块，可以看到平台预置的模型，平台预置了多款等模型，包括文本类、语音类和视觉类，可以直接开通调用。不同的模型的参数和能力不同，我们将持续推出不同能力方向的模型。

模型回答内容重复如何处理?

请求模型的重复惩罚度参数设置过高，会导致重复输出概率增大。建议将重复惩罚度参数设置为0。

最佳实践

零门槛CV大模型训推

概述

本最佳实践面向企业开发者、AI 算法工程师、行业解决方案厂商，基于中国电信星辰 MaaS 模型服务平台，提供一套零门槛、全流程、可视化的 CV（计算机视觉）大模型训练与推理落地方法。使用者无需精通底层 AI 框架与算力运维，无需编写代码，即可快速完成CV 模型创建、数据集准备、自动化调优、服务部署、在线推理全链路操作，实现图像分类、物体检测、图像分割、OCR、视频分析等计算机视觉场景的快速落地。

方案优势

通过本实践，可显著降低 CV 大模型的使用门槛与技术成本，缩短模型从训练到上线的周期，满足政企客户在安防、工业质检、智慧零售、智慧交通、媒体内容生产等行业的视觉 AI 能力快速构建需求。

前置条件

注意

本方案仅作为实践演示，具体环境以用户实际需求为准。

执行本文操作之前， 请完成以下准备工作：

- 注册天翼云账号，并完成实名认证。
- 天翼云账户余额需要大于100元。

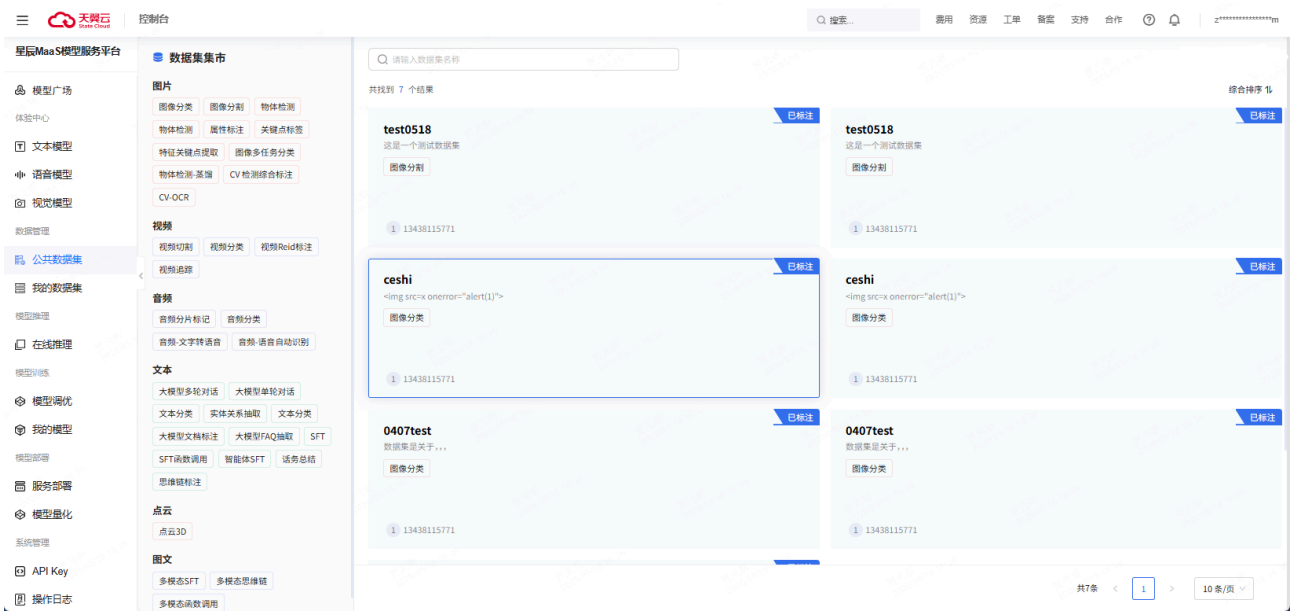
实践步骤

登录星辰MaaS模型服务平台

访问模型推理服务地址： 模型推理服务

准备训练与测试数据集

在左侧菜单栏进入“公共数据集”，通过检索或从侧边栏菜单中选择合适CV的数据集。



创建CV大模型

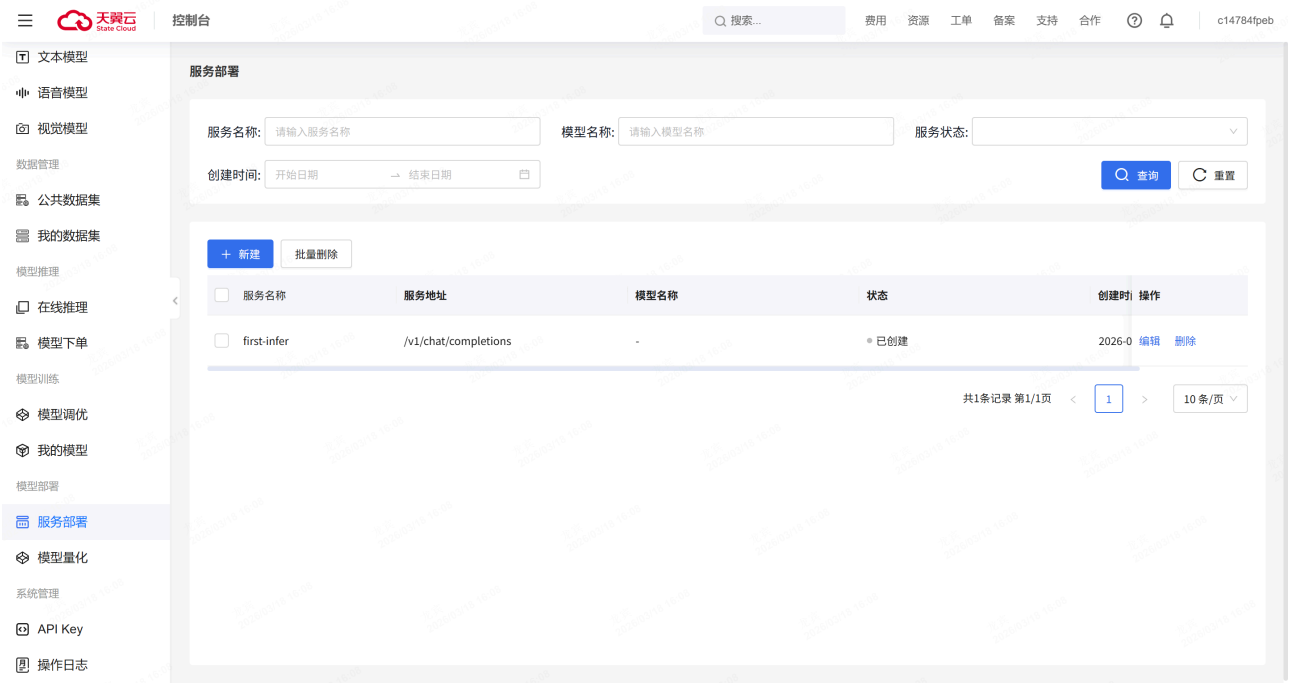
通过“我的模型”模块，通过上传本地模型文件，进行CV模型创建。

进行模型调优

通过“模型调优”模型，新建待调优的模型，并将训练所用的数据集进行匹配，配置好模型调优任务后，启动任务进行自动化调优，调优后可将调优好的模型进行发布操作。

模型部署与调用

调优好的模型，可以通过部署服务，进行模型推理，通过“服务部署“模块，新建推理服务任务，创建成功后，可以通过服务地址进行模型调用。



API参考

开发指南

开发指南

接口信息

API Path

签名认证方式

详细说明：

鉴权目标

本鉴权目的在于为星辰MaaS模型服务平台的鉴权做增强鉴权认证体系，为了系统提供更加高效安全的通信方式。

调用地址

公网域名：xirang-openapi.ctyun.cn 端口：443 接口地址：查看每个接口文档的头部连接地址

请求地址：https://xirang-openapi.ctyun.cn:443/{接口地址}

请求头说明

请求头Header传入如下字段，字段说明见下表格

字段名	是否必填	字段说明
Authorization	是	签名
X-APP-ID	是	用户申请的账号ID
OrderNum	否	订单号

Authorization生成方式

Authorization={originName}/{X-APP-ID}/{region}/{timestamp}/{expirationPeriodInSeconds}/{signedHeaders}/{signature} 其中

- originName 公网环境为：teleai-cloud-auth-v1
- region 所请求服务资源所在的区域 ，可使用省份缩写拼音（见本文后面的省份列表）
- timestamp 为10位的unix 时间锁 例如：1728892448
- expirationPeriodInSeconds签名有效期限从timestamp所指定的时间开始计算，单位为秒
- signedHeaders：推荐默认值直接给【x-app-id】

- signature： 256位签名的十六进制表示，由64个小写字母组成。它由SK(X-APP-KEY)和authStringPrefix哈希得到signingKey，再将canonicalRequest以signingKey为key进行哈希摘要生成，具体算法见下。

附录：

省份拼音缩写表

编码	省市	缩写
000000	全国	QG
110000	北京市	BJ
120000	天津市	TJ
130000	河北省	HE
140000	山西省	SX
150000	内蒙古	NM
210000	辽宁省	LN
220000	吉林省	JL
230000	黑龙江	HL
310000	上海市	SH
320000	江苏省	JS
330000	浙江省	ZJ
340000	安徽省	AH
350000	福建省	FJ
360000	江西省	JX
370000	山东省	SD
410000	河南省	HA
420000	湖北省	HB
430000	湖南省	HN
440000	广东省	GD
450000	广西省	GX
460000	海南省	HI
500000	重庆市	CQ
510000	四川省	SC
520000	贵州省	GZ
530000	云南省	YN
540000	西藏	XZ
610000	陕西省	SN
620000	甘肃省	GS
630000	青海省	QH

编码	省市	缩写
640000	宁夏	NX
650000	新疆	XJ
710000	台湾省	TW
810000	香港	HK
820000	澳门	MO

代码示例

java生成签名工具类

```
package cn.chinatelecom.teleai.utils;

import cn.hutool.core.util.StrUtil;
import lombok.extern.slf4j.Slf4j;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.*;



@Slf4j
public class SignUtils {



    String str ;

    // ########
    public static final int AUTHORIZATION_ITEM_NUM = 7;
    // HMAC SHA256 ####
    private static final String HMAC_SHA256_ALGORITHM = "HmacSHA256";

    public static final List<String> DEFAULT_HEADERS = Arrays.asList(
            HttpHeaders.HOST,
            HttpHeaders.CONTENT_TYPE,
            HttpHeaders.CONTENT_LENGTH,
            "Content-Md5");

    /**
     * ####
     * @param signPrefix  ####
     * @param appKey  ####
     * @param method  HTTP####
     * @param path    HTTP##uri
     * @param params  HTTP####
     * @param headers HTTP##headers
     * @return #####
     */
    public static String genSignature(String signPrefix,String appKey,
 HttpMethod method, String path,
                                      Map<String, String> params, Map<String,
 String> headers,String signedHeaders) {

        // #########
```

```
        String signingKey;
        try {
            // ######
            signingKey = generateHmacSHA256Hex(signPrefix, appKey);
        } catch (Exception e) {
            log.error("{signPrefix}/{X-APP-ID}/{region}/{timestamp}/{expirati
onPeriodInSeconds}####", e);
            throw new RuntimeException(e);
        }
        log.info("appKey:[{}] signingKey:[{}]",appKey,signingKey);
        // ############
        String canonicalRequest = getCanonicalRequest(method, path, params,
 headers,signedHeaders);
        log.info("######## canonicalRequest:###{}",canonicalRequest);
        String signature;
        try {
            // ######
            signature = generateHmacSHA256Hex(canonicalRequest, signingKey);
        } catch (Exception e) {
            log.error("canonicalRequest ####", e);
            throw new RuntimeException(e);
        }
        return signature;
    }

    /**
     * ############
     *
     * @param method  HTTP####
     * @param path    HTTP##uri
     * @param params  HTTP####
     * @param headers HTTP##headers
     * @param signedHeaders ###
     * @return #########
     */
    private static String getCanonicalRequest(HttpMethod method, String path,
 Map<String, String> params,
                                              Map<String, String> headers,
 String signedHeaders) {
        // ############
        StringBuilder canonicalRequest = new StringBuilder();
        canonicalRequest.append(method).append("###");
        String canonicalURI;
        try {
            // #####URI
            canonicalURI = URLEncoder.encode(path, "utf-8").replaceAll("%2F",
 "/");
        } catch (UnsupportedEncodingException e) {
            log.error("url encode error", e);
            throw new RuntimeException(e);
        }
        canonicalRequest.append(canonicalURI).append("###");

        // ##########
        canonicalRequest.append(getCanonicalQueryString(params)).append("##
#");

        // ########
        String canonicalHeaders = getCanonicalHeaders(headers,signedHeaders);
        canonicalRequest.append(canonicalHeaders);
        return canonicalRequest.toString();
    }

    /**
     * ############
     *
     * @param paramMap ######
     * @return #########
```

```
     */
    private static String getCanonicalQueryString(Map<String, String>
 paramMap) {
        List<String> params = new ArrayList<>();
        for (Map.Entry<String, String> param : paramMap.entrySet()) {
            String key = param.getKey();
            String value = param.getValue();
            try {
                params.add(URLEncoder.encode(key, "UTF-8") + "=" + (StrUtil.
isNotBlank(value) ? URLEncoder.encode(value, "UTF-8") : ""));
            } catch (UnsupportedEncodingException e) {
                log.error("params encode error", e);
                throw new RuntimeException(e);
            }
        }
        // ###########
//        params.sort(String.CASE_INSENSITIVE_ORDER);
        Collections.sort(params);
        return StrUtil.join( "&",params);
    }

    /**
     * #########
     * @param headerMap #####
     * @param signedHeaders ###
     * @return ##########
     */
    private static String getCanonicalHeaders(Map<String, String> headerMap,
 String signedHeaders) {
        if (headerMap.isEmpty()) {
            return "";
        }
        // ###########
        Map<String, String> lowerCaseHeaderMap = new HashMap<>();
        for (Map.Entry<String, String> entry : headerMap.entrySet()) {
            lowerCaseHeaderMap.put(entry.getKey().toLowerCase(),
 entry.getValue());
        }

        List<String> headerList = new ArrayList<>();
        List<String> headerNames;
        if (StrUtil.isNotBlank(signedHeaders)) {
            headerNames = StrUtil.split(signedHeaders,";");
        } else {
            headerNames = DEFAULT_HEADERS;
        }
        for (String headerName : headerNames) {
            String lowerCaseHeaderName = headerName.trim().toLowerCase();
            String headerValue = lowerCaseHeaderMap.get(lowerCaseHeaderName);
            if (StrUtil.isNotBlank(headerValue)) {
                try {
                    headerList.add(lowerCaseHeaderName + ":" + StrUtil.trim(
URLEncoder.encode(headerValue, "UTF-8")));
                } catch (UnsupportedEncodingException e) {
                    log.error("headers encode error", e);
                    throw new RuntimeException(e);
                }
            }
        }
        // ##########
//        headerList.sort(String.CASE_INSENSITIVE_ORDER);
        Collections.sort(headerList);
        return StrUtil.join("###",headerList );
    }

    /**
     * #### HmacSHA256 ####
     *
```

```
     * @param data ###########
     * @param key  ##########
     * @return ### HmacSHA256 ##############
     * @throws Exception ###############
     */
    private static String generateHmacSHA256Hex(String data, String key)
 throws Exception {
        SecretKeySpec signingKey = new SecretKeySpec(key.getBytes(), HMAC_
SHA256_ALGORITHM);
        Mac mac = Mac.getInstance(HMAC_SHA256_ALGORITHM);
        mac.init(signingKey);
        byte[] rawHmac = mac.doFinal(data.getBytes());
        return bytesToHex(rawHmac);
    }

    /**
     * ################
     *
     * @param bytes #########
     * @return ############
     */
    private static String bytesToHex(byte[] bytes) {
        StringBuilder hexString = new StringBuilder();
        for (byte b : bytes) {
            String hex = Integer.toHexString(0xff & b);
            if (hex.length() == 1) {
                hexString.append(\'0\');
            }
            hexString.append(hex);
        }
        return hexString.toString();
    }

    public static void main(String[] args) {
        //###################
        //###
        long l = System.currentTimeMillis() / 1000L;
        //########Request ####header#map
        Map<String, String> headerMap = new HashMap<>();
        //################################ x-app-id#
        String signedheaders = "x-app-id";
        //##signedheaders  #############  ### ####request###### ##
signedheaders##key#header#####
        headerMap.put("X-APP-ID", "#######ID");
        //########,###################################
        String region="HN";
        String appId="yourAPPID";
        String basic = "teleai-cloud-auth-v1/"+appId+"/"+region+"/"+l
+"/180000";
        //######param  param### #### ##?##### #######request## ###url###
key=value ########
        //######################map
        Map<String,String> params = new HashMap<>();
        String sign = SignUtils.genSignature(
                basic,
                //X-APP-KEY,########key#
                "yourAPPKEY",
                //#######   POST GET # websocket #GET###############
                HttpMethod.POST,
                //#####path
                "/aipaas/cv/v1/image/cyclistsHelmetDetect",
                params,
                headerMap,
                signedheaders);

        System.out.println(sign);
        //##### Authorization ############
```

```
        System.out.println(String.format("Authorization: %s/%s/
%s",basic,signedheaders,sign));
    }
}

~~~~
```

shell脚本

```
#!/bin/bash

# Function to URL encode a string
url_encode() {
    local string="${1}"
    local strlen=${#string}
    local encoded=""

    for (( pos=0; pos<strlen; pos++ )); do
        c=${string:pos:1}
        case "$c" in
            [a-zA-Z0-9.~_-])
                encoded+="$c"
                ;;
            *)
                printf -v encoded \'%s%%%02X\' "$encoded" "\'$c"
                ;;
        esac
    done
    echo "$encoded"
}

# Function to generate HMAC SHA256 signature
generate_hmac_sha256() {
    local data="${1}"
    local key="${2}"
    echo -n "$data" | openssl dgst -sha256 -hmac "$key" | sed \'s/^.* //\'
}

# Function to get canonical query string
get_canonical_query_string() {
    local param_string="$1"
    local param_list=()

    IFS=\'&\' read -ra param_pairs <<< "$param_string"
    for param_pair in "${param_pairs[@]}"; do
        IFS=\'=\' read -ra param <<< "$param_pair"
        key=${param[0]}
        value=${param[1]}
        encoded_key=$(url_encode "$key")
        encoded_value=$(url_encode "$value")
        param_list+=("${encoded_key}=${encoded_value}")
    done
    #echo "param_list: ${param_list[@]}"
    sort_params=($(for i in "${param_list[@]}"; do echo $i; done | sort))
    #echo "sort_params: ${sort_params[@]}"
    query_string=$(IFS="&"; echo "${sort_params[*]}")
    echo "$query_string"
}

# Function to get canonical headers
get_canonical_headers() {
    local header_string="$1"
    local signed_headers="$2"
    local header_list=()
    IFS=\';\' read -ra headers <<< "$signed_headers"
    for header_name in "${headers[@]}"; do
        IFS=\';\' read -ra header_pairs <<< "$header_string"
```

```
        for header_pair in "${header_pairs[@]}"; do
            IFS=\'=\' read -ra kv <<< "$header_pair"
            name=$(echo "${kv[0]}" | tr \'[:upper:]\' \'[:lower:]\')
            value=${kv[1]}
            if [[ "$name" == "$header_name" ]]; then
                hearder_value=$(url_encode "$value")
                header_list+=("${header_name}:${hearder_value}")
            fi
        done
    done
    #echo "header_list: ${header_list[@]}"
    sort_headers=($(for i in "${header_list[@]}"; do echo $i; done | sort))
    #echo "sort_headers: ${sort_headers[@]}"
    printf "%s
" "${sort_headers[@]}"
}

# Function to generate canonical request
get_canonical_request() {
    local method="$1"
    local path="$2"
    local param_string="$3"
    local header_string="$4"
    local signed_headers="$5"
    local canonicalURI=$(url_encode "$path" | sed \'s/%2F///g\')
    local canonicalQueryString=$(get_canonical_query_string "$param_string")
    local canonicalHeaders=$(get_canonical_headers "$header_string"
 "$signed_headers")
    printf "%s
%s
%s
%s" "$method" "$canonicalURI" "$canonicalQueryString" "$canonicalHeaders"
}

# Function to generate the signature
gen_signature() {
    local appkey="$1"
    local method="$2"
    local path="$3"
    local params="$4"
    local headers="$5"
    local prefix="$6"
    local auth_appid="$7"
    local region="$8"
    local timestamp="$9"
    local expiration="${10}"
    local signed_headers="${11}"
    local basic="${prefix}/${auth_appid}/${region}/${timestamp}/${expiration}"
    #echo "basic: $basic"
    local canonical_request=$(get_canonical_request "$method" "$path"
 "$params" "$headers" "$signed_headers")
    #echo "Canonical Request:  $canonical_request"

    local signing_key=$(generate_hmac_sha256 "$basic" "$appkey")
    #echo "signing_key: $signing_key"
    local signature=$(generate_hmac_sha256 "$canonical_request"
 "$signing_key")

    echo "$signature"
}

# Main function to verify the authorization
verify_auth() {
    local authorization="$1"
    local appid="$2"
    local appkey="$3"
    local method="$4"
    local path="$5"
```

```
    local params="$6"
    local headers="$7"
    echo "authorization: $authorization"
    IFS=\'/\' read -ra auth_parts <<< "$authorization"
    prefix="${auth_parts[0]}"
    auth_appid="${auth_parts[1]}"
    region="${auth_parts[2]}"
    timestamp="${auth_parts[3]}"
    expiration="${auth_parts[4]}"
    signed_headers="${auth_parts[5]}"
    signature="${auth_parts[6]}"
    echo "signature: $signature"
    if [[ "$prefix" != "eop-auth-v1" ]]; then
        echo "Invalid prefix: $prefix"
        return 1
    fi

    if [[ "$appid" != "$auth_appid" ]]; then
        echo "AppID mismatch: $auth_appid != $appid"
        return 1
    fi

    current_time=$(date +%s)
    if (( timestamp + expiration < current_time )); then
        echo "Request expired"
        return 1
    fi

    server_signature=$(gen_signature "$appkey" "$method" "$path" "$params"
 "$headers" "$prefix" "$auth_appid" "$region" "$timestamp" "$expiration"
 "$signed_headers")
    echo "server_signature: $server_signature"
    if [[ "$signature" != "$server_signature" ]]; then
        echo "Signature verification failed"
        return 1
    fi

    echo "Signature verification succeeded"
    return 0
}

# Example usage of gen_signature
method="POST"
path="/v1/test"
params="aa=bb"
headers="date=Mon, 27 Apr 2015 16:23:49 +0800;content-type=text/plain;content-
length=8"

prefix="eop-auth-v1"
auth_appid="aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
appkey="bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb"
region="BJ"
timestamp="1729067670"
expiration="1800"
signed_headers="content-length;content-type;host"

signature=$(gen_signature "$appkey" "$method" "$path" "$params"
 "$headers" "$prefix" "$auth_appid" "$region" "$timestamp" "$expiration"
 "$signed_headers")
echo "Generated Signature: $signature"

# # Example usage of verify_auth
authorization="${prefix}/${auth_appid}/${region}/${timestamp}/${expiration}/
${signed_headers}/${signature}"
verify_auth "$authorization" "$auth_appid" "$appkey" "$method" "$path"
 "$params" "$headers"
```

python 脚本

```
import json
import hashlib
import hmac
import time
import urllib.parse

# ##############)X-APP-ID########ID##X-APP-KEY########KEY#
X_APP_ID = \'yourXAppId\'
X_APP_KEY = \'yourXAppKey\'

# URL #####
REGION = \'QG\'
EXPIRATION_IN_SECONDS = \'43200\'  # ######,####12##
TIMESTAMP = str(int(time.time()))  # #####
SIGNED_HEADERS = \'x-app-id\'  # #####
PUBLIC_NETWORK_SING_HEADER = \'teleai-cloud-auth-v1\'  # #################
EMS_SING_HEADER = \'eop-auth-v1\'  # #################

# #####
headers = {
    \'Content-Type\': \'application/json\',
    \'X-APP-ID\': X_APP_ID,
}


class HttpAuth(object):
    # ####
    def normalize(self, string, encoding_slash=True):
        quoted_string = urllib.parse.quote(string, safe=\'~()*!\'/\' if
 encoding_slash else \'~()*!\'\')
        return quoted_string

    # ##Canonical URI
    def generate_canonical_uri(self, url):
        parsed_url = urllib.parse.urlparse(url)
        return \'/\'.join(self.normalize(segment, False) for segment in
 parsed_url.path.split(\'/\'))

    # ##Canonical Headers
    @staticmethod
    def generate_canonical_headers(input_params):
        signed_headers_list = SIGNED_HEADERS.split(\';\')
        signed_headers_set = set(signed_headers_list)
        sorted_headers = sorted((k.lower(), urllib.parse.quote(v.strip(),
 safe=\'\')) for k, v in input_params.items() if
                                k.lower() in signed_headers_set)
        canonical_headers = \'
\'.join(f"{k}:{v}" for k, v in sorted_headers)
        signed_headers_str = \';\'.join(sorted(signed_headers_set))
        print("canonical_headers:", canonical_headers)
        return canonical_headers, signed_headers_str

    # ####
    def generate_signature(self, x_app_id, x_app_key, region, timestamp,
 expiration_in_seconds, method,
                           canonical_uri,
                           canonical_headers, signed_headers_str, canonical_
query_string):
        # #############
        signing_key_str = f"{PUBLIC_NETWORK_SING_HEADER}/{x_app_id}/{region}/
{timestamp}/{expiration_in_seconds}"

        signing_key = hmac.new(bytes(x_app_key, \'utf-8\'), bytes(signing_key_
str, \'utf-8\'), hashlib.sha256).hexdigest()
```

```
        # print(f"signing_key = {signing_key}")
        canonical_request = f"{method.upper()}
{canonical_uri}
{canonical_query_string}
{canonical_headers}"
        print("canonical_request
", canonical_request)
        signature = hmac.new(bytes(signing_key, \'utf-8\'), bytes(canonical_
request, \'utf-8\'), hashlib.sha256).hexdigest()
        authorization = f"{signing_key_str}/{signed_headers_str}/{signature}"
        # print("authorization", authorization)
        return authorization

    # ##Content-Length
    def get_content_length(self, data):
        body = json.dumps(data)
        return str(len(body))

    # #########
    def get_auth(self, url, method, canonical_query_string):
        """
        ###

        :param url: #####url
        :param method: GET/POST##########websocket###GET
        :param canonical_query_string: params###GET##########key=value########
\'\'
        """
        canonical_uri = self.generate_canonical_uri(url)

        canonical_headers, signed_headers_str = self.generate_canonical_
headers(headers)
        authorization = self.generate_signature(X_APP_ID, X_APP_KEY, REGION,
 TIMESTAMP, EXPIRATION_IN_SECONDS,
                                                method, canonical_uri,
 canonical_headers, signed_headers_str, canonical_query_string)
        headers[\'Authorization\'] = authorization
        print(f"authorization = {authorization}")
        return headers


getAuth = HttpAuth()
```

异步调用结果获取

接口信息

在大模型异步调用反馈结果里，如果是生成图片或者语音，会返回对应结果图片或结果语音文件资源地址；这些异步生成结果并且请求接收后返回"requestId"的能力，若需要获取生成结果内容，需要通过本接口获取。

异步请求结果获取目前支持推、拉两种模式：

- 推模式：在发送异步请求时需要传递"callback_url"（接收回调结果的url地址，不同接口字段名可能有细微不同）字段，即接收回调结果的url地址。详见 开发指南 - 异步调用结果获取 - 推模式 -【客户提供接口】大模型异步调用结果回告协议规范。
- 拉模式：通过发送请求时系统返回的"requestId"字段，主动调用该接口，以实时查询异步调用的结果。

警告

注意：查询大模型生成结果信息接口仅支持查询已经生成的结果，并且生成结果数据在有效期内。

结果未生成或生成结果数据已过期时，接口会返回【992008】获取信息失败；

存在调用结果时，接口返回的信息同各个算法的回告结构。

API Path

/aipaas/lm/v1/asyncResult/query

服务鉴权

服务接口调用时需要严格遵循服务鉴权规则，服务调用鉴权规则请参见： 开发指南 。

请求协议

HTTP

请求方法

GET

请求头部：

头部标签	必填	说明	类型	数据字典	限制	头部内容	示例
Content-Type	是	application/json	[string]	-	-	application/json	application/json
X-APP-ID	是	系统管理--API Key，创建应用获取AppID 和AppKey，公网鉴权，公网调用时必传	[string]	-	-	-	-
Authorization	是	公网鉴权，公网调用时必传	[string]	-	-	-	-
Device-Uuid	否	设备管理-设备uuid	[string]	-	-	-	-

Query参数：

参数名	类型	必填	说明
requestId	[string]	是	请求ID，调用异步算法时返回

请求参数示例：

```
/aipaas/lm/v1/asyncResult/query?requestId=yourRequestId
```

响应内容：

- 数据格式：json
- 外层数据结构：object

参数名	类型	必填
code	[int]	是
message	[string]	是

状态码说明

状态码	返回信息	状态码描述
090003	失败	系统异常
090004	失败	获取信息失败

警告

通用状态码请参考【状态码】中的【网关认证】

调用示例

Java版本

```
import cn.hutool.http.HttpRequest;
import cn.hutool.http.HttpResponse;

import java.util.HashMap;
import java.util.Map;

/** #######HTTP####### */
public class Example {

    public static void main(String[] args) {
        example();
    }

    /**
     * #######HttpRequest#HttpResponse###Hutool####
     * ##maven####
     * <dependency>
     *     <groupId>cn.hutool</groupId>
     *     <artifactId>hutool-all</artifactId>
     *     <version>5.8.29</version>
     * </dependency>
     */
    public static void example() {
        try {
            String url = "######";
            String param = "yourResource";
            // #####
            Map<String, String> headers = new HashMap<>();
            //####
```

```
            headers.put("Content-Type", "application/json");
            headers.put("X-APP-ID", "yourAppId");
            headers.put("Authorization", "yourAuthorization");
            // ##HTTP##
            HttpResponse response =
                    HttpRequest.get(url + param)
                            .execute();
            // ######
            System.out.println(response.body());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

大模型异步调用结果回查

接口信息

API Path

/aipaas/lm/v1/asyncResult/query

请求协议

HTTP

请求方法

GET

请求头部：

头部标签	必填	说明	类型	数据字典	限制	头部内容	示例
Content-Type	是	application/json	[string]	-	-	application/json	application/json
X-APP-ID	是	系统管理--API Key，创建应用获取AppID 和AppKey，公网鉴权，公网调用时必传	[string]	-	-	-	-
Authorization	是	公网鉴权，公网调用时必传	[string]	-	-	-	-
Device-Uuid	否	设备管理-设备uuid	[string]	-	-	-	-

返回结果

Query参数：

参数名	说明	必填	类型	数据字典	限制	示例
requestId	请求ID，调用异步算法时返回	是	[string]	-	-	-

响应内容

参数名	说明	必填	类型	数据字典	限制	示例
code	返回状态码	是	[int]	-	-	-
message	具体信息	是	[string]	-	-	-

成功示例[Mock API]：

```
#####String### ######################
```

失败示例[Mock API]：

```
{
    "code": 992008,
    "message": "######"
}
```

功能简介

用于查询异步算法调用的结果信息，只适用于异步生成结果并且请求接收后返回"requestId"的能力。目前具体支持模型请见下文第6节。 异步请求结果获取目前支持推、拉两种模式：

- 推模式：在发送异步请求时需要传递“callback_url”（接收回调结果的url地址，不同接口字段名可能有细微不同）字段，即接收回调结果的url地址。详见 开发指南 - 异步调用结果获取 - 推模式 -【客户提供接口】大模型异步调用结果回告协议规范。
- 拉模式：通过发送请求时系统返回的"requestId"字段，主动调用该接口，以实时查询异步调用的结果。

查询大模型生成结果信息接口仅支持查询已经生成的结果，并且生成结果数据在有效期内。 结果未生成或生成结果数据已过期时，接口会返回【992008】获取信息失败； 存在调用结果时，接口返回的信息同各个算法的回告结构。

服务鉴权

服务接口调用时需要严格遵循服务鉴权规则，服务调用鉴权规则请参见：开发指南 - 签名认证方式。

状态码说明

返回编码	返回信息	说明
990003	失败	系统异常
992008	失败	获取信息失败

通用状态码请参考【状态码】中的【网关认证】

请求参数示例

```
/aipaas/lm/v1/asyncResult/query?requestId=yourRequestId
```

支持模型

- 文生图大模型
- 图生图大模型

调用示例

Java版本

```
import cn.hutool.http.HttpRequest;
import cn.hutool.http.HttpResponse;

import java.util.HashMap;
import java.util.Map;

/** #######HTTP####### */
public class Example {

    public static void main(String[] args) {
        example();
    }

    /**
     * #######HttpRequest#HttpResponse###Hutool####
     * ##maven####
     * <dependency>
     *     <groupId>cn.hutool</groupId>
     *     <artifactId>hutool-all</artifactId>
     *     <version>5.8.29</version>
     * </dependency>
     */
    public static void example() {
        try {
            String url = "######";
            String requestId = "yourRequestId";
            // #####
            Map<String, String> headers = new HashMap<>();
            //####
            headers.put("Content-Type", "application/json");
            headers.put("X-APP-ID", "yourAppId");
            headers.put("Authorization", "yourAuthorization");
            // ##HTTP##
            HttpResponse response =
                    HttpRequest.get(url)
                               .form("requestId", requestId)
                               .execute();
            // ######
            System.out.println(response.body());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

状态码说明

通用状态码表格

状态码	说明
10011000	请求体超过阈值

状态码	说明
10011001	请求数太多，请稍后再试
10011002	header [Device-Uuid]不能为空
10011003	header [X_APP_ID]不能为空
10011004	header [X_APP_KEY]不能为空
10011005	header [Authorization]不能为空
10011006	header [Authorization] 签名格式不正确
10011007	header [Authorization] 签名前缀不正确
10011008	header [Authorization] appId 与 header appId 不一致
10011009	header [Authorization] 签名已过期
10011010	header [Authorization] 签名认证不通过
10011011	X_APP_ID:在黑名单中，禁止访问
10011012	X_APP_ID,X_APP_KEY校验不通过
10011013	X_APP_ID: 无路径访问权限
10011014	订单: 未开始或者已过期
20001	请求参数安全围栏校验不通过
20002	响应参数安全围栏校验不通过

实时超自然语音合成-普通话版

能力介绍

产品介绍

能够支持将流式文字输入实时合成为播报音频，提供多风格多音色超拟人的音色可供选择，可灵活配置音频参数。

主要功能

- 提供多场景音库： 提供数十种风格多样的音色可供选择，适用于政企、生活、娱乐等多种应用场景。
- 支持灵活播报设置： 支持多种参数配置，可根据场景需求对语音播报的语速、音调、音量进行灵活设置，满足个性化需求。
- 支持流式合成播报： 支持文本流式输入，实时合成音频进行播报。

接口文档

接口信息

API Path

/aipaas/voice/v1/tts/supernaturalrt

请求协议

WS

建立连接时的请求参数（websocket open）：

请求头部：

头部标签	必填	说明	类型	数据字典	限制	头部内容	示例
Content-Type	是	application/json	[string]	-	-	application/json	application/json
X-APP-ID	是	系统管理--API Key，创建应用获取AppID 和AppKey，公网鉴权，公网调用时必传	[string]	-	-	-	-
Device-Uuid	否	设备管理-设备uuid	[string]	-	-	-	-
Authorization	是	鉴权信息	[string]	-	-	-	-

请求参数

Json Object

参数名	说明	必填	类型	数据字典	限制	示例
req_id	请求全局唯一ID，记录该值便于排查问题	是	[string]	-	-	-
text	待合成的文本，需要为UTF-8 编码	是	[string]	-	-	-
format	音频编码格式，支持 PCM格式，默认值：PCM	否	[string]	-	-	-
sample_rate	音频采样率，支持24000Hz、22050Hz、16000 Hz 和8000 Hz，默认值：24000	否	[int]	-		-

参数名	说明	必填	类型	数据字典	限制	示例
voice	说话人，支持音色cixingnan、songchinan、surennan、cixingnan2、zhubonan、kefunan、zhuchinan、jilunan2、surennv、xianliaonv、mansunv、kefunv1、huoponv、ruyanv、xinwennv、kefunv2、luolinv、huoponv2、qingtiannv、cixingnan3、gaojinv、gushinan2、jilunan1、kejinan1、kejinan2、pingdannv、zhiyin、tuokouxiu、xuanyinan、sarah，默认音色为cixingnan	否	[string]	-	-	-
speech_rate	语速，取值范围：[0.5, 2.0]，默认值：1.0	否	[float]	-	-	-
volume	音量，取值范围：[0, 100]，默认值：50	否	[int]	-	-	-
响应报文：	-	-	-	-	-	-

返回结果

成功 (200) Json Object

参数名	说明	必填	类型	数据字典	限制	示例
status	状态码，与code相同（注：2025年11月30日后不再返回此字段）	是	[int]	-	-	-
status_msg	状态说明，与message相同（注：2025年11月30日后不再返回此字段）	是	[string]	-	-	-
code	状态码，见服务码说明	是	[int]	-	-	-
message	状态说明，见服务码说明	是	[string]	-	-	-
sid	会话全局唯一id，用于记录本次会话	是	[string]	-	-	-
result	合成结果	是	[object]	-	-	-
result>>audio	合成音频数据，经过base64 编码	是	[string]	-	-	-
result>>audio_len	合成音频长度，单位：ms	是	[int]	-	-	-
result>>is_end	标志位。true：最后一个合成片段；false：中间合成片段	是	[boolean]	-	-	-

能力简介

超自然流式语音合成提供将输入文本合成为语音二进制数据的功能。

- 支持输出格式：PCM 编码；
- 支持设置采样率：8000 Hz，16000 Hz， 22050Hz， 24000Hz；
- 音频为单声道，位深为 16 bit；
- 支持设置语速、音量；
- 支持设置多种说话人；
- 支持一次性合成 500 字符以内的文字，其中 1 个汉字、1 个英文字母、1 个标点或 1 个句子中间空格均算作 1 个字符，超过 500 个字符的内容将会截断，超过500字的文字不再进行合成；

- 仅支持采用 UTF-8 编码的文本输入；

服务鉴权

服务接口调用时需要严格遵循服务鉴权规则，服务调用鉴权规则请参见：开发指南 - 接口签名认证。

响应结果说明

握手返回结果

返回码	说明	错误信息	解决方法
101	成功	{"message":"success"}	成功，开始语音合成
4001	签名校验失败，授权失败	{"message":"check sign fail"}	联系商务，更新授权
4002	并发请求过多	{"message":"Too many requests."}	联系商务，增加并发

响应结果说明

开始合成响应示例

```
{
   "code": 10000,
   "message": "Success",
   "sid": "4eHgiLCBhbGdvcml"
}
```

接收合成数据响应示例

```
{
    "code": 10000,
    "message": "Success",
    "result": {
        "audio": "//6YYmNiDlSUXjZlDP8yADQAQ1E2ANKJOaTqFGUgIwDQAKAA==",
        "audio_len": 2500,
        "is_end": false
    }
}
```

合成结束响应示例

```
{
    "code": 10000,
    "message": "Success",
    "result": {
        "audio": "//6YYmNiDlSUXjZlDP8yADQAQ1E2ANKJOaTqFGUgIwDQAKAA==",
        "audio_len": 2500,
        "is_end": true
    }
}
```

请求示例

```
{
```

```
    "req_id": "3a87fe9793c9-4ebd-95d4-4ce2-a80c054b",
    "text": "############"
}
```

状态码说明

状态码	解释	说明	解决方法
10301	Parameter error	参数错误	检查请求体是否符合接口协议
10302	Too many requests	并发请求过多	联系商务，增加并发
10304	Parse request body fail	请求格式错误	查看请求的 URL body 格式是否正确，参考接口文档
10503	Server connection time out	服务连接超时	联系技术人员
10903	Synthesis failed	合成失败	联系技术人员
10000	Success	成功	执行下一步操作

超多方言实时语音识别

能力介绍

产品介绍

国内首家单模型同时支持中文、英文及60种方言自由混说，实现“一个”ASR可识别全国多个方言。

主要功能

- 支持中英及60种方言混说识别： 支持中、英、粤语、四川话、重庆话、上海话、武汉话等60+方言的混说转写识别。
- 文本顺滑： 对识别结果进行验证和修正，包括纠错、格式调整等，提高文本的准确性和可读性。
- 标点智能添加： 根据语音内容理解，自动插入适当的标点符号，使转录文本更流畅、更符合书面表达习惯。
- 逆文本标准化： 可讲口语化信息转换为书面语形式，比如数字、日期等信息，方便用户阅读。
- 支持方言热词定制： 用户能够自定义上传的热词，通过优先匹配热词提升在相关领域的识别准确性。

接口文档

接口信息

API Path

/aipaas/voice/v1/asr/fy

请求协议

WS

建立连接时的请求参数（websocket open）：

请求头部：

头部标签	必填	说明	类型	数据字典	限制	头部内容	示例
Content-Type	是	application/json	[string]	-	-	application/json	application/json
X-APP-ID	是	系统管理--API Key，创建应用获取AppID 和AppKey，公网鉴权，公网调用时必传	[string]	-	-	-	-
Device-Uuid	否	设备管理-设备uuid	[string]	-	-	-	-
Authorization	是	鉴权信息	[string]	-	-	-	-

请求参数

Json Object

参数名	说明	必填	类型	数据字典	限制	示例
option	语音识别配置可选项，客户端发送开始识别请求时，根据具体需求，配置该字段	否	[object]	-	-	-
option>>sample_rate	音频采样率，默认值16000Hz	否	[int]	-	-	-
option>>enable_punctuation	是否加标点，默认：是，若开启，则在rec_status为3时给识别文本增加标点	否	[boolean]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
option>>enableinversetext_normalization	是否开启ITN，默认：是，若开启，则在rec_status为3时对识别文本进行逆文本规范化	否	[boolean]	-	-	-
option>>enable_punctuation	是否开启校勘，默认：否，若开启，则在rec_status为3时对识别文本进行校准	否	[boolean]	-	-	-
option>>enable_words	是否开启返回词信息，默认：否，若开启，则同时返回字级别时间戳	否	[boolean]	-	-	-
option>>enable_sc	是否返回繁体中文，默认值是 false	否	[boolean]	-	-	-
option>>provider	（V3.2.0删除）仅对来自电信万号的业务开放（预留字段）	否	[string]	-	-	-
option>>hotwords	热词列表，string数组	否	[array]	-	-	-
option>>hotwordlistid	热词列表查询ID（在热词服务创建热词表后得到热词列表查询ID，ASR服务使用此ID查询生效）	否	[string]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
option>>bannedwordlist_id	敏感词列表查询ID（在敏感词服务创建敏感词列表后得到敏感词列表查询ID，ASR服务使用此ID查询生效）	否	[string]	-	-	-
option>>maxend_silence	句尾静音阈值，单位ms（仅当服务使用普通vad时生效）	否	[int]	-	-	-
option>>format	音频格式，支持pcm、opus，默认值是pcm，传入其它值会报错（opus格式需要明确是ogg封装的opus格式）	否	[string]	-	-	-
req_id	请求全局唯一id，记录该值便于排查问题【发送开始识别rec_status=0时必填】	是	[string]	-	-	-
rec_status	识别状态 0：开始识别；1：发送语音流； 2：结束语音流；	是	[int]	-	-	-
audio_stream	发送语音流时必填。语音流，采用base64 编码	否	[string]	-	-	-
响应报文：	-	-	-	-	-	-

返回结果

成功 (200) Json Object

参数名	说明	必填	类型	数据字典	限制	示例
code	返回码	是	[int]	-	-	10000
message	返回码描述	是	[string]	-	-	success
sid	会话全局唯一id，用于记录本次会话	是	[string]	-	-	aae36140-bc13-441f-81f9-6700fe7a5e96
res_status	响应状态 0：识别就绪；1：识别到有效语音开始；2：如果开启了返回中间结果， 则返回中间识别结果；3：检测到一段有效语音结束，返回该段语音的识别结果； 4：处理完所有的音频数据，返回尚未返回的识别结果（如果有）；	是	[int]	-	-	2
elps_time	当前识别结果所对应的已处理的音频总时长，单位是毫秒	否	[int]	-	-	-
data	识别结果，服务端接收到语音流后返回	是	[object]	-	-	-
data>>sn	句子编号，从0开始	否	[int]	-	-	-
data>>results	当前句子识别结果，如果开启object.nbest，则返回多个结果	否	[array]	-	-	-
data>>results>>text	句子识别结果	否	[string]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
data>>results>>begin_time	句子开始时间，单位是毫秒	否	[int]	-	-	-
data>>results>>end_time	句子结束时间，单位是毫秒	否	[int]	-	-	-
data>>results>>loudness	句子音量，仅在一句话结束即res_status=3时返回	否	[float]	-	-	-
data>>results>>speech_rate	句子语速，仅在一句话结束即res_status=3时返回	否	[float]	-	-	-
data>>results>>pitch	句子语调，仅在一句话结束即res_status=3时返回	否	[float]	-	-	-
data>>results>>lang	当前方言种类，仅在Sensevoice模型二刷时有效，仅在状态3有效	否	[string]	-	-	-
data>>results>>words	当前句子的词信息	否	[array]	-	-	-
data>>results>>words>>text	词信息	否	[string]	-	-	-
data>>results>>words>>begin_time	词开始时间，单位是毫秒	否	[int]	-	-	-
data>>results>>words>>end_time	词结束时间，单位是毫秒	否	[int]	-	-	-

能力简介

多方言实时语音识别是指能够同时处理多种方言的语音识别系统。这种系统可以实现在不同方言之间进行实时转换和识别，广泛应用于多语言环境下的语音交互场景。

- 音频属性：采样率8k/16k/其他常见采样率，位宽 16 bit，单声道
- 音频格式：PCM 音频流
- 字符编码：UTF-8
- 响应格式：统一采用 JSON 格式

- 音频格式：PCM音频流，0GG封装的opus音频流

服务鉴权

服务接口调用时需要严格遵循服务鉴权规则，服务调用鉴权规则请参见：开发指南 - 签名认证方式。

鉴权状态码

code	说明	错误描述信息	解决方法
101	成功	{"message":"success"}	成功，开始语音识别
4002	并发超过限制	{"message":"server overflow"}	联系商务，增加并发
4004	授权失败	{"message":"invalid license"}	联系运维人员生成有效license

请求示例

开始识别

客户端发送开始识别请求，需要通过请求body带语音识别过程中的可选配置参数，示例：

```
{
    "option": {
        "sample_rate": 16000,
        "enable_punctuation": true,
        "enable_inverse_text_normalization": true
    },
    "req_id": "aae36140-bc13-441f-81f9-6700fe7a5e96",
    "rec_status": 0
}
```

响应结果

```
{
    "code": 10000,
    "message": "success",
    "sid": "aae36140-bc13-441f-81f9-6700fe7a5e96",
    "res_status": 0
}
```

发送语音流

客户端接收到服务端发送的确认识别请求有效的响应后，开始发送语音流数据，请求body各参数含义如下：

名称	类型	必需	说明
rec_status	int	是	识别状态0：开始识别；1：发送语音流；2：结束语音流；
audio_stream	string	是	语音流，采用 base64 编码

示例：

```
{
 "rec_status":1,
 "audio_stream":"000asraae361406700fe7a5e9681f956210b5f1270"
}
```

接收识别结果

客户端接收到服务端发送的确认检测请求有效的响应后，开始持续接收识别结果。

```
{
    "code": 10000,
    "message": "success",
    "sid": "aae36140-bc13-441f-81f9-6700fe7a5e96",
    "res_status": 2,
    "data": {
        "sn": 1,
        "results": [{
            "lang": "zh",
            "text": "####",
            "begin_time": 1500,
            "end_time": 2800,
            "words": [
                {
                    "text": "#",
                    "begin_time": 50,
                    "end_time": 70
                },
                {
                    "text": "#",
                    "begin_time": 50,
                    "end_time": 70
                },
                {
                    "text": "#",
                    "begin_time": 50,
                    "end_time": 70
                },
                {
                    "text": "#",
                    "begin_time": 50,
                    "end_time": 70
                }
            ]
        }]
    }
}
```

结束语音流

客户端语音流发送完成，结束语音流，请求body各参数含义如下：

名称	类型	必需	说明
rec_status	int	是	识别状态0：开始识别；1：发送语音流；2：结束语音流；

示例：

```
{
    "rec_status": 2
```

```
}
```

关闭连接

客户端如果不需要继续进行语音识别，则立即关闭websocket 连接（避免占用资源），如果需要继续进行语音识别(多轮对话场景)，需要从开始识别状态开始，按照上述步骤依次执行

状态码说明

状态码	解释	说明	解决方法
10000	success	成功	执行下一步操作
20003	Banned word(s) detected in input	敏感词命中	成功，并检测到敏感词
10001	parse request body fail	URL body 格式不对	查看请求的 URL body 格式是否正确，参考接口文档
10002	session not found	会话id查询失败	检查客户端发送的请求，通常是因为没有发送开始识别请求
10003	required parameter miss	参数缺失	检查接口文档，补全入参
10004	duplicated session id	会话id重复	检查客户端发送的请求，通常是因为重复发送开始识别请求
10005	worker pool overflow	超并发	联系研发人员进行排查
10006	unknown error	未知错误	联系研发人员进行排查
10007	Non-real-time audio data	非实时音频数据	检查发送的音频数据是否与每次发送间隔的时间一致,比如每200ms发送200ms的音频数据
10008	Session not begun	尚未发送开始识别标志	检查是否发送开始识别标志
10009	Session is running	重复发送开始识别标志	检查是否重复发送开始识别标志，若重复，请先发送结束识别标志
10010	Hotword list load failed	热词表查询失败	检查X-APP-ID，热词表ID是否有错误，检查热词查询服务是否有错误
10011	Banned Word List Loads failed	敏感词表查询失败	检查X-APP-ID，敏感词ID是否有错误，检查敏感词查询服务是否有错误

通用问答版大模型-36B

能力介绍

产品介绍

通用版大模型36B，给用户提供文本生成、文本理解、逻辑推理、数学计算等全场景通用的问答大模型服务能力。

主要功能

TeleChat3-36B-v1.1.0算法能力属于通用问答大模型，其具备文本理解、文本创作生成、逻辑推理、语言翻译、数学计算等多种场景的问答能力。

产品定价

算法名称	报价类型	报价
通用版大模型36B	标准报价	共享资源:2200元/并发・月

警告

注意：如返回网关错误码（见开发指南->公共常量->返回状态码），则本次调用不做计费。

接口文档

接口信息

API Path

/aipaas/lm/v1/telechat/ty36b

请求协议

HTTP

请求方法

POST

请求头部：

头部标签	必填	说明	类型	数据字典	限制	头部内容	示例
Content-Type	是	application/json;UTF-8	[string]	-	-	application/json	application/json

头部标签	必填	说明	类型	数据字典	限制	头部内容	示例
X-APP-ID	是	系统管理--API Key，创建应用获取AppID 和AppKey，公网鉴权，公网调用时必传	[string]	-	-	-	-
Device-Uuid	否	设备管理-设备uuid	[string]	-	-	-	-
Authorization	是	鉴权信息	[string]	-	-	-	-

请求参数

Json Object

参数名	说明	必填	类型	数据字典	限制	示例
messages	对话内容。多轮对话场景下，完整的content长度不可超过模型最大输入。（聊天中content可为空）	是	[array]	-	-	-
messages>>content	string/array，消息的内容	是	[string]	-	-	-
messages>>refusal	助理的拒绝消息	否	[string]	-	-	-
messages>>role	消息作者的角色	是	[string]	-	-	-
messages>>name	参与者的可选名称。提供模型信息以区分同 一角色的参与者，最大长度为 30	否	[string]	-	-	-
messages>>tool_calls	模型生成的工具调用	否	[array]	-	-	-
messages>>tool_calls>>id	工具调用的 ID	是	[string]	-	-	-
messages>>tool_calls>>type	工具的类型。【目前仅支持 function，未严格校验】	是	[string]	-	-	-
messages>>tool_calls>>function	模型调用的函数。	是	[object]	-	-	-
messages>>tool_calls>>function>>name	要调用的函数的名称	是	[string]	-	-	-
messages>>tool_calls>>function>>arguments	string/obj，用于调用函数的参数，由模型生成。【标准是string类型，推荐使用string】	是	[string]	-	-	-
messages>>toolcallid	此消息正在响应的工具调用，最大长度 为 64	否	[string]	-	-	-
model	要使用的模型的 ID(Chat)，最大长度为 64	是	[string]	-	-	-
maxcompletiontokens	最大生成词元数量。输入词元和生成词元的总长度受模型上下文长度的限制。兼容传入 max_tokens，但不允许两个参数同时存在，范围在 1 到模型最大输出长度（例如：2000）	否	[int]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
presence_penalty	默认值为 0,在 -2.0 到 2.0 之间。正值会基于新生成词是否已经出现在当前文本中进行惩罚，增加模型谈论新主题的可能性。	否	[number]	-	-	-
stream	默认值为 false，如果设置true，回答内容将使用流式传输。	否	[boolean]	-	-	-
stream_options	流式响应的选项。仅当您设置stream: true时设置此项才生效。 "streamoptions": {"includeusage": true} 如果设置，则在data: [DONE]之前流式传输的一个块中usage此块上的字段显示整个请求的token使用情况统计信息。	否	[object]	-	-	-
tools	模型可以调用的工具列表【模型中工具传入，在生成时候不一定会调用】	否	[array]	-	-	-
tools>>type	工具的类型。目前仅支持 function	是	[string]	-	-	-
tools>>function	模型调用的函数	是	[object]	-	-	-
tools>>function>>description	函数用途的描述，模型使用它来选择何时以 及如何调用函数，最大长度 2000	否	[string]	-	-	-
tools>>function>>name	要调用的函数的名称。必须是 a-z、A-Z、0-9 或包含下划线和短划线，最大长度为 64	是	[string]	-	-	-
tools>>function>>parameters	函数接受的参数，描述为 JSON Schema 对象。请参阅 指南 有关示例，请参阅 JSON 架构参考 有关格式的文档	否	[object]	-	-	-
tools>>function>>strict	是否在生成函数调用时启用严格的架构遵 循。如果设置为 true，则模型将遵循字段中定义的确切架构。当 is 时，仅支持 JSON 架构的子集。在函数调用指南中了解有关结构化输出的更多信息	否	[boolean]	-	-	-
tool_choice	控制模型调用工具。 none 表示模型不会调 用任何工具，而是生成一条消息。 auto 表示模型可以在生成消息或调用一个或多个工具之间进行选择	否	[string]	-	-	-
temperature	默认值为 0.3，使用的采样温度，范围 [0,2]。较高的值如 0.8 会使输出更随机，而较低的值如 0.2 会使输出更集中且具有确定性，取 0 时会关闭采样，无随机输出。	否	[number]	-	-	-
top_p	默认值为 1，取值范围为[0,1]之间，作为温 度采样的替代方法，称为核采样，该方法中模型考虑具有 top_p 概率质量的词元结果	否	[number]	-	-	-
extra_body	额外参数	否	[object]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
extrabody>>topk	作用：随机候选数（不建议修改） 取值：大于0 的值越大，输出的多元性越大。 取值范围：(0,10000]	否	[int]	-	-	-
extrabody>>repetitionpenalty	作用：生成重复惩罚 取值范围： （0,2]	否	[double]	-	-	-
extrabody>>frequencypenalty	基于它们在生成文本中的频率。 值>0 鼓励模型使用新令牌，而值<0 鼓励模型重复令牌。取值范围：[-2,0)(0,2]	否	[double]	-	-	-
traceId	接口跟踪 ID，每次调用需不同，可使用 UUID，用于日志排除、链路跟踪，最大长度为64	否	[string]	-	-	-
timestamp	当前时间戳【秒级】	否	[int]	-	-	-

响应内容：

返回结果

流式响应结果-成功 (200) Json Object

参数名	说明	必填	类型	数据字典	限制	示例
data	表示流式返回的数据块，包含生成的文本内容，[DONE]表示生成过程已完成	是	[object]	-	-	-
data>>id	聊天补全的唯一标识符(uuid4)。	是	[string]	-	-	-
data>>choices	聊天补全选项的列表。	是	[array]	-	-	-
data>>choices>>finish_reason	string/null，模型停止生成标记的原因。原因可能是stop模型到达自然停止点或提供的停止序列、 length达到请求中指定的最大标记数。	否	[string]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
data>>choices>>index	选择列表中的索引	否	[int]	-	-	-
data>>choices>>delta	模型响应流生成的聊天补全增量。	否	[object]	-	-	-
data>>choices>>delta>>content	string/null，块消息的内容。	否	[string]	-	-	-
data>>choices>>delta>>role	此消息作者的角色。【第一块携带、其他块不携带此参数】	否	[string]	-	-	-
data>>choices>>delta>>refusal	string/null，模型生成的拒绝消息【第一块携带、其他块不携带此参数】	否	[string]	-	-	-
data>>choices>>delta>>tool_calls	工具列表	否	[array]	-	-	-
data>>choices>>delta>>tool_calls>>index	选择列表中的索引	否	[int]	-	-	-
data>>choices>>delta>>tool_calls>>id	工具调用的ID	否	[string]	-	-	-
data>>choices>>delta>>tool_calls>>type	工具的类型。目前仅支持function	否	[string]	-	-	-
data>>choices>>delta>>tool_calls>>function	模型调用的函数。	否	[object]	-	-	-
data>>choices>>delta>>tool_calls>>function>>name	要调用的函数的名称	否	[string]	-	-	-
data>>choices>>delta>>tool_calls>>function>>arguments	用于调用函数的参数，由模型以 JSON 格式生成。请注意，该模型并不总是生成有效的 JSON，并且可能会产生函数架构未定义的参数。在调用函数之前验证代码中的参数	否	[string]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
data>>choices>>logprobs	object/null，输出标记的对数概率。	否	[object]	-	-	-
data>>created	时间戳【秒级】	是	[int]	-	-	-
data>>model	用于生成补全的模型。	是	[string]	-	-	-
data>>object	对象类型，始终为 chat.completion.chunk。	是	[string]	-	-	-
data>>usage	object/null，token统计【当stream_options生效时，流式最后一个块携带 usage 信息，其他情况默认不携带】	否	[object]	-	-	-
data>>usage>>completion_tokens	生成token	否	[int]	-	-	-
data>>usage>>prompt_tokens	prompt的tokens	否	[int]	-	-	-
data>>usage>>total_tokens	总token	否	[int]	-	-	-
data>>usage>>completiontokensdetails	生成 tokens详细统计	否	[object]	-	-	-
data>>usage>>completiontokensdetails>>reasoning_tokens	思考 tokens，当前都为 0	否	[int]	-	-	-
data>>system_fingerprint	代表模型运行时后端配置的指纹。【当前默认null】	是	[string]	-	-	-

非流式响应结果-成功 (200) Json Object

参数名	说明	必填	类型	数据字典	限制	示例
id	聊天补全的唯一标识符(uuid4)。	是	[string]	-	-	-
choices	聊天补全选项的列表。	是	[array]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
choices>>finish_reason	模型停止生成标记的原因。原因可能是stop模型到达自然停止点或提供的停止序列、length达到请求中指定的最大标记数。 如果模型调用了工具，tool_calls。	是	[string]	-	-	-
choices>>index	选择列表中的索引	是	[int]	-	-	-
choices>>message	模型生成的聊天补全消息。	是	[object]	-	-	-
choices>>message>>content	消息的内容	是	[string]	-	-	-
choices>>message>>refusal	模型生成的拒绝消息。	是	[string]	-	-	-
choices>>message>>tool_calls	模型生成的工具调用，例如函数调用。	否	[array]	-	-	-
choices>>message>>tool_calls>>id	工具调用的ID。	否	[string]	-	-	-
choices>>message>>tool_calls>>type	工具的类型。目前仅支持function。	否	[string]	-	-	-
choices>>message>>tool_calls>>function	模型调用的函数。	否	[object]	-	-	-
choices>>message>>tool_calls>>function>>name	要调用的函数的名称。	否	[string]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
choices>>message>>tool_calls>>function>>arguments	用于调用函数的参数，由模型以 JSON 格式生成。请注意，该模型并不总是生成有效的 JSON，并且可能会产生函数架构未定义的参数。在调用函数之前验证代码中的参数。	是	[string]	-	-	-
choices>>message>>role	此消息作者的角色。	是	[string]	-	-	-
choices>>logprobs	object/null，输出标记的对数概率。	是	[object]	-	-	-
created	时间戳【秒级】	是	[int]	-	-	-
model	用于聊天补全的模型。	是	[string]	-	-	-
object	对象类型，始终为 chat.completion。	是	[string]	-	-	chat.completion.chunk
usage	token统计	是	[object]	-	-	-
usage>>completion_tokens	生成tokens	是	[int]	-	-	-
usage>>prompt_tokens	prompt的tokens	是	[int]	-	-	-
usage>>total_tokens	总tokens	是	[int]	-	-	-
usage>>completiontokensdetails	生成 tokens 详细统计	是	[object]	-	-	-
usage>>completiontokensdetails>>reasoning_tokens	思考 tokens，当前都为 0	是	[int]	-	-	-
system_fingerprint	代表模型运行时后端配置的指纹。【当前默认null】	是	[string]	-	-	-

成功示例[Mock API]：

```
data: {
    "id": "277fd1f21c9e4b15bbbc6f28b0fc62ab",
    "choices": [{
```

```
        "finish_reason": null,
        "index": 0,
        "delta": {
            "content": "",
            "role": "assistant",
            "refusal": null
        },
        "logprobs": null
    }],
    "created": 1729666765,
    "system_fingerprint": null,
    "model": "aichat",
    "object": "chat.completion.chunk"
}
data: {
        "id": "277fd1f21c9e4b15bbbc6f28b0fc62ab",
        "choices": [{
            "finish_reason": null,
            "index": 0,
            "delta": {
                "content": "##"
            },
            "logprobs": null
        }],
        "created": 1729666765,
        "model": "aichat",
        "system_fingerprint": null,
        "object": "chat.completion.chunk"
    }
    ........#######
data: {
    "id": "277fd1f21c9e4b15bbbc6f28b0fc62ab",
    "choices": [{
        "finish_reason": null,
        "index": 0,
        "delta": {
            "content": "#"
        },
        "logprobs": null
    }],
    "created": 1729666765,
    "model": "aichat",
    "system_fingerprint": null,
    "object": "chat.completion.chunk"
}
data: {
    "id": "277fd1f21c9e4b15bbbc6f28b0fc62ab",
    "choices": [{
        "finish_reason": "stop",
        "index": 0,
        "delta": {},
        "logprobs": null
    }],
    "created": 1729666765,
    "model": "aichat",
    "system_fingerprint": null,
    "object": "chat.completion.chunk"
}
data: {
    "id": "c86aff51e90442ed8c7e1ab5296189a8",
    "choices": [],
    "created": 1729666822,
    "model": "aichat",
    "system_fingerprint": null,
    "object": "chat.completion.chunk",
    "usage": {
        "completion_tokens": 40,
        "prompt_tokens": 23,
```

```
        "total_tokens": 63,
        "completion_tokens_details": {
            "reasoning_tokens": 0
        }
    }
}
#####stream_option#####
data: [DONE]
```

功能介绍

适用星辰MaaS模型服务平台API通用服务，给用户提供文本生成、文本理解、逻辑推理、数学计算等全场景通用的问答大模型服务能力。其具备文本理解、文本创作生成、逻辑推理、语言翻译、数学计算等多种场景的问答能力。

服务鉴权

服务接口调用时需要严格遵循服务鉴权规则，服务调用鉴权规则请参见：开发指南 - 签名认证方式。

状态码说明

通用状态码请参考【状态码】中的【网关认证】

字段	类型	说明
code	string	状态响应码。01110001： 请求体参数不符合类型；01110002： 请求参数不符合规则；01110003： 模型连接错误；01110004： 内部错误
message	string	错误消息提示

请求参数示例

请求示例

流式请求示例

```
{
    "model": "Chat",
    "messages": [{
        "role": "user",
        "content": "###############"
    }],
    "max_completion_tokens": 150,
    "stream": true
}
```

流式响应示例

```
data: {
    "id": "277fd1f21c9e4b15bbbc6f28b0fc62ab",
    "choices": [{
        "finish_reason": null,
        "index": 0,
        "delta": {
            "content": "",
            "role": "assistant",
            "refusal": null
```

```
        },
        "logprobs": null
    }],
    "created": 1729666765,
    "system_fingerprint": null,
    "model": "aichat",
    "object": "chat.completion.chunk"
}
data: {
        "id": "277fd1f21c9e4b15bbbc6f28b0fc62ab",
        "choices": [{
            "finish_reason": null,
            "index": 0,
            "delta": {
                "content": "u4f60u597d"
            },
            "logprobs": null
        }],
        "created": 1729666765,
        "model": "aichat",
        "system_fingerprint": null,
        "object": "chat.completion.chunk"
    }
    ........#######
data: {
    "id": "277fd1f21c9e4b15bbbc6f28b0fc62ab",
    "choices": [{
        "finish_reason": null,
        "index": 0,
        "delta": {
            "content": "uff1f"
        },
        "logprobs": null
    }],
    "created": 1729666765,
    "model": "aichat",
    "system_fingerprint": null,
    "object": "chat.completion.chunk"
}
data: {
    "id": "277fd1f21c9e4b15bbbc6f28b0fc62ab",
    "choices": [{
        "finish_reason": "stop",
        "index": 0,
        "delta": {},
        "logprobs": null
    }],
    "created": 1729666765,
    "model": "aichat",
    "system_fingerprint": null,
    "object": "chat.completion.chunk"
}
data: {
    "id": "c86aff51e90442ed8c7e1ab5296189a8",
    "choices": [],
    "created": 1729666822,
    "model": "aichat",
    "system_fingerprint": null,
    "object": "chat.completion.chunk",
    "usage": {
        "completion_tokens": 40,
        "prompt_tokens": 23,
        "total_tokens": 63,
        "completion_tokens_details": {
            "reasoning_tokens": 0
        }
    }
}
```

```
##### stream_option #####data: [DONE]
```

非流式请求示例

```
{
    "model": "Chat",
    "messages": [{
        "role": "user",
        "content": "###############"
    }],
    "max_completion_tokens": 150,
    "stream": false
}
```

非流式响应示例

```
{
    "id": "aacd0a837f244378812fa898d5094fe2",
    "choices": [{
        "finish_reason": "stop",
        "index": 0,
        "message": {
            "refusal": null,
            "role": "assistant",
            "content": " ############################"
        },
        "logprobs": null
    }],
    "created": 1729666692,
    "model": "aichat",
    "system_fingerprint": null,
    "object": "chat.completion",
    "usage": {
        "completion_tokens": 41,
        "prompt_tokens": 23,
        "total_tokens": 64,
        "completion_tokens_details": {
            "reasoning_tokens": 0
        }
    }
}
```

健康监测接口协议

- 请求方法：GET
- 访问URL：http://ip:port/api/v2/Health

返回参数	数据类型	说明	备注
code	int	状态码	无
data	obj	返回消息体	无
data.error_code	string	返回服务状态码	无
data.error_message	string	返回模型状态信息	无
data.algo_types	array	算法类型	-

响应示例：

```
{
    "code": 10000,
    "data": {
```

```
        "error_code": "APP_ERR_OK",
        "error_message": "####",
        "algo_types": ["0000"]
    }
}
```

注意：模型健康检测接口http_code返回非200，即可认为模型异常。

message参数说明

当message角色为system时

字段	必填	类型	说明
content	是	string/array	系统消息的内容 例：“content”:”你好” “content”: [{“type”:”text”,”text”:”你好”}]
role	是	string	消息作者的角色，在本例中为 system
name	否	string	参与者的可选名称。提供模型信息以区分同一角色的参与者，最大长度为 30

当message角色为user时

字段	必填	类型	说明
content	是	string/array	用户消息的内容。
role	是	string	消息作者的角色，在本例中为 user
name	否	string	参与者的可选名称。提供模型信息以区分同一角色的参与者，最大长度为 30

当message角色为assistant时

字段	必填	类型	说明
content	是	string/array	用户消息的内容。
refusal	否	string	助理的拒绝消息。
role	是	string	消息作者的角色，在本例中assistant
name	否	string	参与者的可选名称。提供模型信息以区分同一角色的参与者，最大长度为 30
tool_calls	否	array	模型生成的工具调用。详细见请求字段

当message角色为tool时

字段	必填	类型	说明
role	是	string	消息作者的角色，在本例中为tool
content	是	string/array	工具消息的内容。
toolcallid	否	string	此消息正在响应的工具调用，最大长度为 64

调用示例

Java版本

```
import cn.hutool.http.HttpRequest;
import cn.hutool.http.HttpResponse;
import cn.hutool.json.JSONUtil;

import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

/** #######HTTP####### */
public class Example {

    public static void main(String[] args) {
        example();
    }

    /**
     * ####### JSONUtil#HttpRequest#HttpResponse###Hutool####
     * ##maven####
     * <dependency>
     *     <groupId>cn.hutool</groupId>
     *     <artifactId>hutool-all</artifactId>
     *     <version>5.8.29</version>
     * </dependency>
     */
    public static void example() {
        try {
            String url = "######";
            // #####
            Map<String, String> headers = new HashMap<>();
            //####
            headers.put("Content-Type", "application/json");
            headers.put("X-APP-ID", "yourAppId");
            headers.put("Authorization", "yourAuthorization");

            // ######
            Map<String, Object> request = new HashMap<>();
            request.put("model", "Chat");
            Map<String, Object> messages = new HashMap<>();
            messages.put("role", "user");
            messages.put("content", "##############");
            request.put("messages", Arrays.asList(messages));
            request.put("max_completion_tokens", 150);
            request.put("stream", true);
            // ########JsonNode
            String requestString = JSONUtil.toJsonStr(request);

            // ##HTTP##
            HttpResponse response =
                    HttpRequest.post(url)
```

```
                            .headerMap(headers, true)
                            .body(requestString)
                            .execute();
            // ######
            System.out.println(response.body());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

Python版本

```
import json
import hashlib
import hmac
import time
import re
import urllib.parse
import requests
import warnings
from datetime import datetime
import logging

# ############INFO########
logging.basicConfig(level=logging.INFO, format=\'%(asctime)s - %(levelname)s -
 %(message)s\')

# ######
X_APP_ID = "yourAppId"
AUTHORIZATION = "yourAuthorization"

# URL #####
url = "######"  # ###URL
request_data = {
    "model": "Chat",
    "messages": [{
        "role": "user",
        "content": "###############"
    }],
    "max_completion_tokens": 150,
    "stream": True
}

# #####
headers = {
    "Content-Type": "application/json",
    "X-APP-ID": X_APP_ID,
    "Authorization": AUTHORIZATION
}


def timeSimple(timestamp):
    # #######HH:MM:SS########
    dt_object = datetime.fromtimestamp(timestamp)
    formatted_time = dt_object.strftime("%H:%M:%S")
    return formatted_time


def send_request(url):
    try:
        start_time = time.time()
        logging.info(f"####: {url}")
        logging.info(f"####: {timeSimple(start_time)}")

        with requests.post(
```

```
            url, json=request_data, headers=headers, stream=True, verify=False
        ) as response:
            first_packet_time = None
            if response.status_code == 200:
                logging.info(f"#####: {timeSimple(time.time())}")

                for chunk in response.iter_content(chunk_size=1024):
                    if chunk:
                        if first_packet_time is None:
                            first_packet_time = time.time()
                        logging.info(
                            f"Received chunk: {timeSimple(time.time())}
 {chunk.decode(\'utf-8\')}"
                        )

                end_time = time.time()
                logging.info(f"Time to first byte (TTFB): {first_packet_time -
 start_time:.3f} seconds")
                logging.info(f"Request completed in {end_time -
 start_time:.3f} seconds")
            else:
                logging.error(f"Request failed with status code {response.
status_code}")
    except Exception as e:
        logging.error(f"An error occurred: {e}")


# ####
send_request(url)
logging.info(f"headers = {headers}")
```

对话大模型

能力介绍

产品介绍

对话大模型能力是指系统能够处理用户的多样化输入和精准输出对应回答，实现类似人类之间自然交流的“边问边答” 能力。通过大规模预训练、知识图谱融合、语义理解与推理等技术，实现更高效、智能的人机交互。消除传统问答系统的生硬与局限，保证对话的流畅、深度与灵活。

主要功能

- 实时双向交互：用户与模型可同时进行交流，如同日常对话般顺畅，不再受传统问答系统一问一答的单向模式限制，极大提升交互效率和体验。
- 灵活打断与引导：用户能随时插话、改变话题或调整提问方向，模型会及时响应并给出合适回答，交互更加自然，无需等待一轮对话结束。
- 多轮对话连贯：具备强大的上下文理解能力，可记住用户前几轮对话内容，在后续交流中提供连贯且有针对性的回答，保证对话的连贯性和逻辑性。
- 多形式反馈同步：在用户输入问题时，模型不仅能快速给出答案，还可根据需求实时呈现相关的知识解释、参考资料等内容，提高交互的可视化和信息丰富度。

接口文档

接口信息

API Path

/aipaas/lm/v1/telechat/completions115

请求协议

HTTP

请求方法

POST

请求头部：

头部标签	必填	说明	类型	数据字典	限制	头部内容	示例
Content-Type	是	application/json	[string]	-	-	-	-
X-APP-ID	是	系统管理--API Key，创建应用获取AppID 和AppKey，公网鉴权，公网调用时必传	[string]	-	-	-	-
Order-Num	是	订单编号	[string]	-	-	-	-
Authorization	是	鉴权信息	[string]	-	-	-	-

请求参数 Json Object

参数名	说明	必填	类型	数据字典	限制	示例
messages	对话内容	是	[array]	-	-	-
messages>>content	系统消息的内容string/array类型，例：“content”：“你好” “content”：[{“type”：”text”,”text”：”你好”}]	是	[object]	-	-	-
messages>>refusal	助理的拒绝消息	否	[string]	-	-	-
messages>>role	消息作者的角色	是	[string]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
messages>>name	参与者的可选名称	否	[string]	-	-	-
messages>>toolcallid	此消息正在响应的工具调用	否	[string]	-	-	-
messages>>tool_calls	模型生成的工具调用	否	[array]	-	-	-
model	要使用的模型的ID(当前为telechat-115b)，最大长度为 64	是	[string]	-	-	-
maxcompletiontokens	最大生成词元数量。输入词元和生成词元的总长度受模型上下文长度的限制。兼容传入max_tokens，但不允许两个参数同时存在，范围在1到模型最大输出长度（例如：2000）	是	[int]	-	-	-
presence_penalty	默认值为 0,在 -2.0 到 2.0 之间。正值会基于新生成词是否已经出现在当前文本中进行惩罚，增加模型谈论新主题的可能性。	否	[number]	-	-	-
stream	默认值为false，如果设置true，回答内容讲使用流式传输	否	[boolean]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
stream_options	流式响应的选项。仅当您设置stream: true时设置此项才生效。"streamoptions": {"includeusage": true} 如果设置，则在data: [DONE]之前流式传输的一个块中usage此块上的字段显示整个请求的token使用情况统计信息，所有其他块也将包含一个usage字段，但值为空。	否	[object]	-	-	-
tools	模型可以调用的工具列表。	否	[array]	-	-	-
tools>>type	工具的类型。目前仅支持function	是	[string]	-	-	-
tools>>function	模型调用的函数	是	[object]	-	-	-
tools>>function>>description	函数用途的描述，模型使用它来选择何时以及如何调用函数，最大长度2000	否	[string]	-	-	-
tools>>function>>name	要调用的函数的名称。必须是 a-z、A-Z、0-9或包含下划线和短划线，最大长度为 64	是	[string]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
tools>>function>>parameters	函数接受的参数，描述为 JSON Schema 对象。请参阅指南 有关示例，请参阅 JSON 架构参考 有关格式的文档	否	[object]	-	-	-
tools>>function>>strict	是否在生成函数调用时启用严格的架构遵循。如果设置为 true，则模型将遵循字段中定义的确切架构。当 is 时，仅支持 JSON 架构的子集。在函数调用指南中了解有关结构化输出的更多信息	否	[boolean]	-	-	-
tool_choice	控制模型调用工具。 none表示模型不会调用任何工具，而是生成一条消息。 auto表示模型可以在生成消息或调用一个或多个工具之间进行选择	否	[string]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
temperature	默认值为0.3，使用的采样温度，范围 [0,2]。较高的值如 0.8 会使输出更随机，而较低的值如 0.2 会使输出更集中且具有确定性，取0时会关闭采样，无随机输出。	否	[number]	-	-	-
top_p	默认值为 1，取值范围为[0,1]之间，作为温度采样的替代方法，称为核采样，该方法中模型考虑具有 top_p 概率质量的词元结果	否	[number]	-	-	-
extra_body	额外参数	否	[object]	-	-	-
extrabody>>topk	作用：随机候选数（不建议修改） 取值：大于0的值越大，输出的多元性越大。取值范围：(0,10000]	否	[int]	-	-	-
extrabody>>repetitionpenalty	默认值为1.01，作用：生成重复惩罚（不建议修改） 取值范围：（0,2]	否	[double]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
extrabody>>frequency_penalty	默认值为0，基于它们在生成文本中的频率。值>0鼓励模型使用新令牌，而值<0鼓励模型重复令牌。 TODO:值阈	否	[double]	-	-	-
traceId	接口跟踪ID，每次调用需不同，可使用UUID， 用于日志排除、链路跟踪，最大长度为 64	否	[string]	-	-	-
timestamp	当前时间戳	否	[long]	-	-	-
knowledgeBaseId	知识库编码	是	[string]	-	-	-

响应内容：

返回结果

流式响应结果-成功 (200) Json Object

参数名	说明	必填	类型	数据字典	限制	示例
data	表示流式返回的数据块，包含生成的文本内容，[DONE]表示生成过程已完成	是	[object]	-	-	-
data>>id	聊天补全的唯一标识符(uuid4)。	是	[object]	-	-	-
data>>choices	聊天补全选项的列表。	是	[array]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
data>>choices>>finish_reason	模型停止生成标记的原因。原因可能是stop模型到达自然停止点或提供的停止序列、 length达到请求中指定的最大标记数。	是	[string]	-	-	-
data>>choices>>index	选择列表中的索引	是	[int]	-	-	-
data>>choices>>delta	模型响应流生成的聊天补全增量。	是	[object]	-	-	-
data>>choices>>delta>>content	块消息的内容。	是	[string]	-	-	-
data>>choices>>delta>>role	此消息作者的角色。【第一块携带，其他块不携带此参数】	是	[string]	-	-	-
data>>choices>>delta>>refusal	模型生成的拒绝消息	是	[string]	-	-	-
data>>choices>>delta>>tool_calls	工具列表	是	[array]	-	-	-
data>>choices>>delta>>tool_calls>>index	选择列表中的索引	是	[int]	-	-	-
data>>choices>>delta>>tool_calls>>id	工具调用的ID	是	[string]	-	-	-
data>>choices>>delta>>tool_calls>>type	工具的类型。目前仅支持function	是	[string]	-	-	-
data>>choices>>delta>>tool_calls>>function	模型调用的函数。	是	[object]	-	-	-
data>>choices>>delta>>tool_calls>>function>>name	要调用的函数的名称	是	[string]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
data>>choices>>delta>>tool_calls>>function>>arguments	用于调用函数的参数，由模型以 JSON 格式生成。请注意，该模型并不总是生成有效的 JSON，并且可能会产生函数架构未定义的参数。在调用函数之前验证代码中的参数	是	[string]	-	-	-
data>>choices>>logprobs	输出标记的对数概率。	是	[object]	-	-	-
data>>created	时间戳	是	[int]	-	-	-
data>>model	用于生成补全的模型。	是	[string]	-	-	-
data>>object	对象类型，始终为 chat.completion.chunk。	是	[string]	-	-	-
data>>usage	token统计【当stream_options生效时，最后一个块携带usage信息、其他块携带的是null，其他情况默认不携带】	是	[object]	-	-	-
data>>usage>>completion_tokens	生成token	是	[int]	-	-	-
data>>usage>>prompt_tokens	prompt的tokens	是	[int]	-	-	-
data>>usage>>total_tokens	总token	是	[int]	-	-	-
data>>usage>>completiontokensdetails	忽略，该算法未使用	是	[object]	-	-	-
data>>usage>>completiontokensdetails>>reasoning_tokens	忽略，该算法未使用	是	[int]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
data>>system_fingerprint	代表模型运行时后端配置的指纹。可以与请求参数seed 结合使用，以了解可能影响确定性的后端更改何时发生。【当前默认null】	是	[string]	-	-	-

非流式响应结果-成功 (200) Json Object

参数名	说明	必填	类型	数据字典	限制	示例
id	聊天补全的唯一标识符(uuid4)。	是	[string]	-	-	-
choices	聊天补全选项的列表。	是	[array]	-	-	-
choices>>finish_reason	模型停止生成标记的原因。原因可能是stop模型到达自然停止点或提供的停止序列、length达到请求中指定的最大标记数。 如果模型调用了工具，tool_calls。	是	[string]	-	-	-
choices>>index	选择列表中的索引	是	[int]	-	-	-
choices>>message	模型生成的聊天补全消息。	是	[object]	-	-	-
choices>>message>>content	消息的内容	是	[string]	-	-	-
choices>>message>>refusal	模型生成的拒绝消息。	是	[string]	-	-	-
choices>>message>>tool_calls	模型生成的工具调用，例如函数调用。	是	[array]	-	-	-
choices>>message>>tool_calls>>id	工具调用的ID。	是	[string]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
choices>>message>>tool_calls>>type	工具的类型。目前仅支持function。	是	[string]	-	-	-
choices>>message>>tool_calls>>function	模型调用的函数。	是	[object]	-	-	-
choices>>message>>tool_calls>>function>>name	要调用的函数的名称。	是	[string]	-	-	-
choices>>message>>tool_calls>>function>>arguments	用于调用函数的参数，由模型以 JSON 格式生成。请注意，该模型并不总是生成有效的 JSON，并且可能会产生函数架构未定义的参数。在调用函数之前验证代码中的参数。	是	[string]	-	-	-
choices>>message>>role	此消息作者的角色。	是	[string]	-	-	-
choices>>logprobs	输出标记的对数概率。	是	[object]	-	-	-
created	时间戳	是	[int]	-	-	-
model	用于聊天补全的模型。	是	[string]	-	-	-
object	对象类型，始终为chat.completion。	是	[string]	-	-	chat.completion.chunk
usage	token统计	是	[object]	-	-	-
usage>>completion_tokens	生成 tokens	是	[int]	-	-	-
usage>>prompt_tokens	prompt的tokens	是	[int]	-	-	-
usage>>total_tokens	总tokens	是	[int]	-	-	-
usage>>completiontokensdetails	忽略，该算法未使用	是	[object]	-	-	-
usage>>completiontokensdetails>>reasoning_tokens	忽略，该算法未使用	是	[int]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
system_fingerprint	代表模型运行时后端配置的指纹。可以与请求参数seed 结合使用，以了解可能影响确定性的后端更改何时发生。【当前默认null】	是	[string]	-	-	-

功能介绍

TeleChat对话115B版本。

服务鉴权

服务接口调用时需要严格遵循服务鉴权规则，服务调用鉴权规则请参见：开发指南 - 接口签名认证。

状态码说明

通用状态码请参考【状态码】中的【网关认证】

```
{
    "code": "10010003",
    "message": "#####"
}
```

请求参数示例

流式请求示例

```
{
    "model": "telechat-115b",
    "messages": [{
        "role": "user",
        "content": "###############"
    }],
    "max_completion_tokens": 150,
    "stream": true
}
```

流式响应示例

```
data: {
    "id": "277fd1f21c9e4b15bbbc6f28b0fc62ab",
    "choices": [{
        "finish_reason": null,
        "index": 0,
        "delta": {
            "content": "",
            "role": "assistant",
            "refusal": null
        },
        "logprobs": null
    }],
```

```
    "created": 1729666765,
    "system_fingerprint": null,
    "model": "telechat-115b",
    "object": "chat.completion.chunk"
}

data: {
        "id": "277fd1f21c9e4b15bbbc6f28b0fc62ab",
        "choices": [{
            "finish_reason": null,
            "index": 0,
            "delta": {
                "content": "##"
            },
            "logprobs": null
        }],
        "created": 1729666765,
        "model": "telechat-115b",
        "system_fingerprint": null,
        "object": "chat.completion.chunk"
    }
    ........#######
data: {
    "id": "277fd1f21c9e4b15bbbc6f28b0fc62ab",
    "choices": [{
        "finish_reason": null,
        "index": 0,
        "delta": {
            "content": "#"
        },
        "logprobs": null
    }],
    "created": 1729666765,
    "model": "telechat-115b",
    "system_fingerprint": null,
    "object": "chat.completion.chunk"
}

data: {
    "id": "277fd1f21c9e4b15bbbc6f28b0fc62ab",
    "choices": [{
        "finish_reason": null,
        "index": 0,
        "delta": {
            "content": ""
        },
        "logprobs": null
    }],
    "created": 1729666765,
    "model": "telechat-115b",
    "system_fingerprint": null,
    "object": "chat.completion.chunk"
}

data: {
    "id": "277fd1f21c9e4b15bbbc6f28b0fc62ab",
    "choices": [{
        "finish_reason": "stop",
        "index": 0,
        "delta": {},
        "logprobs": null
    }],
    "created": 1729666765,
    "model": "telechat-115b",
    "system_fingerprint": null,
    "object": "chat.completion.chunk"
}
```

```
data: {
    "id": "c86aff51e90442ed8c7e1ab5296189a8",
    "choices": [],
    "created": 1729666822,
    "model": "telechat-115b",
    "system_fingerprint": null,
    "object": "chat.completion.chunk",
    "usage": {
        "completion_tokens": 40,
        "prompt_tokens": 23,
        "total_tokens": 63,
        "completion_tokens_details": {
            "reasoning_tokens": 0
        }
    }
}
#####stream_option#####
data: [DONE]
```

非流式请求示例

```
{
    "model": "telechat-115b",
    "messages": [{
        "role": "user",
        "content": "###############"
    }],
    "max_completion_tokens": 150,
    "stream": false
}
```

非流式响应示例

```
{
    "id": "chat-fb53de120a09438babb58fe3d8aba2e7",
    "object": "chat.completion",
    "created": 1730964205,
    "model": "opencompass",
    "choices": [{
        "index": 0,
        "message": {
            "role": "assistant",
            "content": "Of course! How can I assist?"
        },
        "tool_calls": [],
        "logprobs": null,
        "finish_reason": "length",
        "stop_reason": null
    }],
    "usage": {
        "prompt_tokens": 34,
        "total_tokens": 41,
        "completion_tokens": 7,
        "prompt_logprobs": null
    }
}
```

message参数说明

当message角色为system时

字段	必填	类型	说明
content	是	string/array	系统消息的内容 例：“content”:”你好” “content”: [{“type”:”text”,”text”:”你好”}]
role	是	string	消息作者的角色，在本例中为 system
name	否	string	参与者的可选名称。提供模型信息以区分同一角色的参与者。

当message角色为user时

字段	必填	类型	说明
content	是	string/array	用户消息的内容。
role	是	string	消息作者的角色，在本例中为 user
name	否	string	参与者的可选名称。提供模型信息以区分同一角色的参与者。

当message角色为assistant时

字段	必填	类型	说明
content	是	string/array	用户消息的内容。
refusal	否	string	助理的拒绝消息。
role	是	string	消息作者的角色，在本例中assistant
name	否	string	参与者的可选名称。提供模型信息以区分同一角色的参与者。
tool_calls	否	array	模型生成的工具调用。

当message角色为tool时

字段	必填	类型	说明
role	是	string	消息作者的角色，在本例中为tool
content	是	string/array	工具消息的内容。

字段	必填	类型	说明
toolcallid	否	string	此消息正在响应的工具调用。

messages列表格式规则

messages 列表是一个包含多个消息对象的数组，每个消息对象包含一个 role 属性，该属性的值为 "system"、"user"、"assistant" 或 "tool" 中的一个。messages 列表必须遵守以下规则：

1. 列表中第一个消息的 role 必须是 "system" 或 "user"，最后一个消息的 role 必须是 "user" 或 "tool"。
2. 如果一个消息的 role 是 "system"，那么它下一个消息的 role 必须是 "user"。
3. 如果一个消息的 role 是 "user"且不是最后一个消息，那么它下一个消息的 role 必须是 "assistant"。
4. 如果一个消息的 role 是 "tool"且不是最后一个消息，那么它下一个消息的 role 必须是 "tool"或"assistant"。
5. 如果一个消息的 role 是 "assistant"，那么它下一个消息的 role 可以是多个"tool" 或一个 "user"。
6. "system"、"user"、"assistant" 不能连续多次出现，即不能有两个连续的消息拥有相同的这些角色。"tool" 可以连续多次出现。

调用示例

Java版本

```
import cn.hutool.http.HttpRequest;
import cn.hutool.http.HttpResponse;
import cn.hutool.json.JSONUtil;

import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

/** #######HTTP####### */
public class Example {

    public static void main(String[] args) {
        example();
    }

        /**
         * ####### JSONUtil#HttpRequest#HttpResponse###Hutool####
         * ##maven####
         *
         *             cn.hutool
         *             hutool-all
         *             5.8.29
         *
         */

    public static void example() {
        try {
            String url = "######";
            // #####,########
            Map headers = new HashMap<>();
            headers.put("X-APP-ID", "yourAppId");
            headers.put("Order-Num", "yourOrderNum");
            headers.put("Authorization", "yourAuthorization");
            // ######
            Map request = new HashMap<>();
            request.put("model", "telechat-115b");
            Map messages = new HashMap<>();
```

```
            messages.put("role", "user");
            messages.put("content", "###############");
            request.put("messages", Arrays.asList(messages));
            request.put("max_completion_tokens", 150);
            request.put("stream", true);
            // ########JsonNode
            String requestString = JSONUtil.toJsonStr(request);

            // ##HTTP##
            HttpResponse response =
                    HttpRequest.post(url)
                            .headerMap(headers, true)
                            .body(requestString)
                            .execute();
            // ######
            System.out.println(response.body());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

Python版本

```
import json
import requests
from typing import Dict, List, Any, Optional

def example():
    """
    ##requests###HTTP#####Java Hutool##
    """
    try:
        url = "######"

        # ###############
        headers = {
            "X-APP-ID": "yourAppId",
            "Order-Num": "yourOrderNum",
            "Authorization": "yourAuthorization"
        }

        # ######
        request = {
            "model": "telechat-115b",
            "messages": [
                {
                    "role": "user",
                    "content": "###############"
                }
            ],
            "max_completion_tokens": 150,
            "stream": True
        }

        # ##HTTP##
        response = requests.post(
            url=url,
            headers=headers,
            json=request,  # #####JSON###Content-Type
            timeout=30
        )

        # ######
        print(response.text)
        return response.text
```

```
    except Exception as e:
        print(f"####: {e}")
        return None


# ####
def main():
    result1 = example()

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

文生图大模型

能力介绍

产品介绍

文生图大模型可通过选择生成写实、水彩水墨、动漫等风格图像，通过语义解析、图像生成、风格迁移等技术，实现从文字创意到视觉呈现的高效转化。用户通过文字描述，能快速获得满足个性化需求的图像作品，显著提升图像创作的灵活性与效率，无需专业设计工具，可通过文本指令动态调整图像内容。

主要功能

- 智能图像生成：基于用户输入的文字描述，精准解析语义信息与创作需求，快速生成符合主题、构图合理、细节丰富的图像。
- 多种风格选择：支持在图像生成过程中，选择艺术风格，如写实、水彩水墨、动漫等，满足多样化审美需求。
- 多轮迭代优化：可根据用户的反馈意见，对已生成图像进行细节优化、元素增减、色彩调整等迭代创作，提供连贯的图像创作体验，逐步完善图像直至达到理想效果。
- 图文同步预览：在图像生成过程中，保留原始文本描述，方便用户对比查看，确认图像与文字描述的契合度，及时提出修改意见，提高图像创作的可视化程度与最终成品的准确性 。

接口文档

接口信息

API Path

/aipaas/lm/v1/text/picture

请求协议

HTTP

请求方法

POST

请求头部：

头部标签	必填	说明	类型	数据字典	限制	头部内容	示例
Content-Type	是	application/json	[string]	-	-	application/json	application/json
X-APP-ID	是	系统管理--API Key，创建应用获取AppID 和AppKey，公网鉴权，公网调用时必传	[string]	-	-	-	-
Authorization	是	公网鉴权，公网调用时必传	[string]	-	-	-	-
Device-Uuid	否	设备管理-设备uuid	[string]	-	-	-	-

请求参数 Json Object

参数名	说明	必填	类型	数据字典	限制	示例
seq_id	请求序列号，请求的唯一标识	是	[string]	-	-	-
task_id	能力编号，当前能力编号为10001	是	[int]	-	-	-
prompt	文本内容	是	[string]	-	-	-
resolution_ratio	分辨率	否	[array]	[1,1]:分辨率1:1,[9,16]:分辨率9:16,[16,9]:分辨率16:9,[4,3]:分辨率4:3	-	-
style	生成风格	否	[int]	0:通用风格,1:写实风格,2:卡通风格,3:3D风格,4:素描风格,5:中国画风格,6:油画风格,7:水彩风格,8:彩铅风格,9:像素风格	-	-

参数名	说明	必填	类型	数据字典	限制	示例
batch_size	生成图片数量，默认4张，最大支持4张。	否	[int]	-	-	-
callback_url	接收回调结果的url地址，return_mode为0是必传	否	[string]	-	-	-
return_mode	结果返回模式，0:push，1:pull，默认为0。push模式即提供callback_url接收最终结果回调；pull模型即通过大模型异步调用结果回查接口查询最终调用结果，接口文档详见开发指南下的大模型异步调用结果回查。	否	[int]	-	-	-

响应内容：

返回结果

成功 (200) Json Object

参数名	说明	必填	类型	数据字典	限制	示例
code	状态响应码。参考错误编码规范	是	[string]	-	-	-
msg	调用结果描述。参考错误编码规范	是	[string]	-	-	-
requestId	请求ID，用于调用查询大模型生成结果信息接口	是	[string]	-	-	-

成功示例[Mock API]：

```
{
    "code": "10000",
```

```
    "msg": "####",
    "requestId": "test001"
}
```

失败示例[Mock API]：

```
{
    "code": "10903",
    "msg": "######",
    "requestId": "test001"
}
```

功能简介

该算法提供根据输入文字和风格制作并生成结果图片的能力。

服务鉴权

服务接口调用时需要严格遵循服务鉴权规则 公网服务调用鉴权规则请参见：开发指南 - 接口签名认证

状态码说明

通用状态码请参见【状态码】中的【网关认证】，其余状态码如下：

返回编码	返回信息	说明
10000	服务执行成功	服务执行成功
10100	算法模型异常	算法模型调用失败(超时)
10101	结果图片不合规	算法生成图片存在不合规元素，不进行输出
10200	图片下载异常	图片 url 访问失败，下载异常
10201	图片格式错误	下载后的图片格式不对，非图片（除 jpeg, png, jpg）
10202	算法推理异常	算法推理失败，生成图片失败
10203	图片上传异常	结果图片上传 ceph/minIO 失败
10204	图片分辨率过小	图片分辨率过小，无法正常生成结果
10205	图片检测到多个人脸	图片检测到多个人脸
10206	图片面部较小	图片检测到脸部过小
10207	人脸仅检测到侧脸	人脸只检测到侧脸
10208	图片不存在人脸	图片不存在人脸
10209	图片姿态不符合要求	人脸朝向不符合要求
10210	图片面部存在遮挡	人脸存在遮挡情况
10211	图片面部模糊	图片人脸过于模糊
10212	图片非全身照	图片中的人物非全身照，可能存在人体部分被遮挡情况
10213	图片脸部不完整	图片脸部不完整

返回编码	返回信息	说明
10301	服务必填参数缺失	服务必填参数缺失或未填写
10304	服务入参错误	部分参数存在限制说明，会针对入参做校验返回；例如传入的 style 风格不在范围内，prompt 入参不合规等情况
10903	服务执行失败	服务执行失败(可能存在组件连接异常、算法模型程序运行异常等情况，导致服务无法正常运行)

请求参数示例

```
{
    "seq_id": "test001",
    "task_id": 10001,
    "prompt": "############",
    "resolution_ratio": [1, 1],
    "style": 2,
    "batch_size": 1,
        "callback_url": "http://your-callback-address"
}
```

请求示例

Java版本

```
import cn.hutool.http.HttpRequest;
import cn.hutool.http.HttpResponse;
import cn.hutool.json.JSONObject;
import cn.hutool.json.JSONUtil;

import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

public class ImageGenerateText2Image {

    public static void main(String[] args) {
        text2ImageRequest();
    }

    /**
     * ####### JSONObject#HttpRequest#HttpResponse###Hutool####
     * ##maven####
     *          <dependency>
     *              <groupId>cn.hutool</groupId>
     *              <artifactId>hutool-all</artifactId>
     *              <version>5.8.29</version>
     *          </dependency>
     */
    public static void text2ImageRequest() {
        String url = "######";
        // #####
        Map<String, String> headers = new HashMap<>();
        //######
        headers.put("X-APP-ID", "yourAppId");
        headers.put("Authorization", "yourAuthorization");
        headers.put("Content-Type", "application/json");
        // ######
```

```
        Map<String, Object> request = new HashMap<>();
        request.put("seq_id", "test001");
        request.put("task_id", 10001);
        request.put("prompt", "############");
        request.put("resolution_ratio", Arrays.asList(1, 1));
        request.put("style", 2);
        request.put("batch_size", 1);
        request.put("return_mode", 1);
        // ########
        String requestString = JSONUtil.toJsonStr(request);
        // ##HTTP##
        HttpResponse response =
                HttpRequest.post(url)
                        .headerMap(headers, false)
                        .body(requestString)
                        .execute();
        // ######
        System.out.println(response.body());
    }
}
```

Python示例

```
import requests
import json
import logging

# ####
logging.basicConfig(level=logging.INFO, format=\'%(asctime)s - %(levelname)s -
 %(message)s\')
logger = logging.getLogger(__name__)

def text_to_image_request_enhanced():
    """
    ##############
    ################
    """
    try:
        url = "######"
        headers = {
            "X-APP-ID": "yourAppId",
            "Authorization": "yourAuthorization",
            "Content-Type": "application/json"
        }

        # #####
        request_body = {
            "seq_id": "test001",
            "task_id": 10001,
            "prompt": "############",
            "resolution_ratio": [1, 1],
            "style": 2,
            "batch_size": 1,
            "return_mode": 1
        }

        logger.info(f"##############: {url}")
        logger.debug(f"###: {headers}")
        logger.debug(f"###: {json.dumps(request_body, indent=2,
 ensure_ascii=False)}")

        # ##POST##
        response = requests.post(
            url=url,
            headers=headers,
            json=request_body,
            timeout=30
```

```
        )

        logger.info(f"#####: {response.status_code}")

        # ####
        if response.status_code == 200:
            try:
                response_data = response.json()
                logger.info("############")
                logger.debug(f"####: {json.dumps(response_data, indent=2,
 ensure_ascii=False)}")

                # ##################URL
                if "image_url" in response_data:
                    logger.info(f"####URL: {response_data[\'image_url\']}")
                elif "image_data" in response_data:
                    logger.info("#######Base64###")

                return response_data

            except json.JSONDecodeError:
                logger.error("#######JSON##")
                logger.error(f"####: {response.text}")
                return None

        else:
            logger.error(f"##############: {response.status_code}")
            logger.error(f"####: {response.text}")
            return None

    except requests.exceptions.Timeout:
        logger.error("#################")
    except requests.exceptions.ConnectionError:
        logger.error("########URL###########")
    except requests.exceptions.HTTPError as e:
        logger.error(f"HTTP##: {e}")
    except Exception as e:
        logger.error(f"######: {e}")

# ########
def save_image_from_response(response_data, output_path):
    """
    ##############
    """
    try:
        if "image_data" in response_data:
            # #######Base64###
            import base64
            image_data = response_data["image_data"]
            if image_data.startswith("data:image"):
                # ##data URL##
                image_data = image_data.split(",")[1]

            with open(output_path, "wb") as f:
                f.write(base64.b64decode(image_data))
            logger.info(f"######: {output_path}")

        elif "image_url" in response_data:
            # ######
            image_url = response_data["image_url"]
            image_response = requests.get(image_url, timeout=30)
            with open(output_path, "wb") as f:
                f.write(image_response.content)
            logger.info(f"####URL######: {output_path}")

        else:
            logger.warning("#############URL")
```

```
    except Exception as e:
        logger.error(f"#######: {e}")

# ###
if __name__ == "__main__":
    result = text_to_image_request_enhanced()
    if result:
        # ####################
        save_image_from_response(result, "generated_image.jpg")
        print("##########")
```

图生图大模型

能力介绍

产品介绍

图生图模型能力是指系统能够同时处理图像输入与图像输出，实现基于源图像快速生成多样化目标图像的智能创作能力。通过图像特征提取、风格迁移、内容重构等技术，实现从原始图像到新图像的高效转化。并可支持多种风格选择。

主要功能

- 实时图像转换：系统可基于用户输入的原始图像，快速生成目标风格或内容的新图像，在转换过程中实现流畅的实时反馈。
- 图像动态调整：用户可对图像转换过程进行干预，通过添加指令、调整参数、增减元素等操作，对生成图像进行实时修改和补充。
- 多轮图像优化：具备图像历史记忆能力，能记住用户前几轮的图像输入和调整指令，基于上下文理解，提供连贯的图像优化体验，根据用户需求逐步完善图像，直至达到理想效果。
- 风格动态切换：用户可根据创作需求，切换写实、水彩水墨、国漫、皮克斯、动漫等多种风格，系统即时响应并呈现对应风格的图像效果。

接口文档

接口信息

API Path /aipaas/lm/v1/picture/picture

请求协议 HTTP

请求方法 POST

请求头部：

头部标签	必填	说明	类型	数据字典	限制	头部内容	示例
Content-Type	是	application/json	[string]	-	-	application/json	application/json

头部标签	必填	说明	类型	数据字典	限制	头部内容	示例
X-APP-ID	是	系统管理--API Key，创建应用获取AppID 和AppKey，公网鉴权，公网调用时必传	[string]	-	-	-	-
Authorization	是	公网鉴权，公网调用时必传	[string]	-	-	-	-
Device-Uuid	否	设备管理-设备uuid	[string]	-	-	-	-

请求参数 Json Object

参数名	说明	必填	类型	数据字典	限制	示例
seq_id	请求序列号，请求的唯一标识	是	[string]	-	-	-
task_id	能力编号，当前能力编号为20001	是	[int]	-	-	-
image	输入图片url	是	[array]	-	-	-
style	生成风格	是	[int]	0:冬日暖阳,1:薰衣草,2:宇航员,3:梅花汉服,4:圣诞（室外）,5:圣诞（室内）,6:烟花盛典,7:街景机甲,8:喜迎新春,9:龙年大吉,10:新春闪耀,11:东北往事,12:时代风华	-	-
batch_size	生成图片数量，目前只支持传1	否	[int]	-	-	-
callback_url	接收回调结果的url地址，return_mode为0是必传	否	[string]	-	-	-

参数名	说明	必填	类型	数据字典	限制	示例
return_mode	结果返回模式，0:push，1:pull，默认为0。push模式即提供callback_url接收最终结果回调；pull模型即通过大模型异步调用结果回查接口查询最终调用结果，接口文档详见开发指南下的大模型异步调用结果回查。	否	[int]	-	-	-

响应内容：

返回结果

成功 (200) Json Object

参数名	说明	必填	类型	数据字典	限制	示例
code	状态响应码。参考错误编码规范	是	[string]	-	-	-
msg	调用结果描述。参考错误编码规范	是	[string]	-	-	-
requestId	请求ID，用于调用查询大模型生成结果信息接口	是	[string]	-	-	-

成功示例[Mock API]：

```
{
    "code": "10000",
    "msg": "####",
    "requestId": "test001"
}
```

失败示例[Mock API]：

```
{
    "code": "10903",
    "msg": "######",
    "requestId": "test001"
}
```

功能简介

该功能提供根据指定风格对输入图片进行加工，并生成加工后图片的能力。

服务鉴权

服务接口调用时需要严格遵循服务鉴权规则 公网服务调用鉴权规则请参见：开发指南 - 接口签名认证

状态码说明

通用状态码请参见【状态码】中的【网关认证】，其余状态码如下：

返回编码	返回信息	说明
10000	服务执行成功	服务执行成功
10100	算法模型异常	算法模型调用失败(超时)
10101	结果图片不合规	算法生成图片存在不合规元素，不进行输出
10200	图片下载异常	图片 url 访问失败，下载异常
10201	图片格式错误	下载后的图片格式不对，非图片（除 jpeg, png, jpg）
10202	算法推理异常	算法推理失败，生成图片失败
10203	图片上传异常	结果图片上传 ceph/minIO 失败
10204	图片分辨率过小	图片分辨率过小，无法正常生成结果
10205	图片检测到多个人脸	图片检测到多个人脸
10206	图片面部较小	图片检测到脸部过小
10207	人脸仅检测到侧脸	人脸只检测到侧脸
10208	图片不存在人脸	图片不存在人脸
10209	图片姿态不符合要求	人脸朝向不符合要求
10210	图片面部存在遮挡	人脸存在遮挡情况
10211	图片面部模糊	图片人脸过于模糊
10212	图片非全身照	图片中的人物非全身照，可能存在人体部分被遮挡情况
10213	图片脸部不完整	图片脸部不完整
10301	服务必填参数缺失	服务必填参数缺失或未填写
10304	服务入参错误	部分参数存在限制说明，会针对入参做校验返回；例如传入的 style 风格不在范围内，prompt 入参不合规等情况
10903	服务执行失败	服务执行失败(可能存在组件连接异常、算法模型程序运行异常等情况，导致服务无法正常运行)

请求参数示例

```
{
    "seq_id": "a504324c-6e10-11e8-adf0-58fb8443ee27",
    "task_id": 20001,
    "image": ["http://xxx"],
    "style": 0,
    "batch_size": 1,
    "callback_url": "http://callbackUrl"
}
```

回告结果

接口回告结果信息详见同级目录下的【回告数据结构】。

调用示例

Java版本

```
import cn.hutool.http.HttpRequest;
import cn.hutool.http.HttpResponse;
import cn.hutool.json.JSONUtil;

import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

/** #######HTTP####### */
public class Example {

    public static void main(String[] args) {
        example();
    }

        /**
         * ####### JSONUtil#HttpRequest#HttpResponse###Hutool####
         * ##maven####
         *         <dependency>
         *             <groupId>cn.hutool</groupId>
         *             <artifactId>hutool-all</artifactId>
         *             <version>5.8.29</version>
         *         </dependency>
         */

    public static void example() {
        try {
            String url = "######";
            // #####
            Map<String, String> headers = new HashMap<>();
            //######
            headers.put("X-APP-ID", "yourAppId");
            headers.put("Authorization", "yourAuthorization");
            headers.put("Content-Type", "application/json");
            // ######
            Map<String, Object> request = new HashMap<>();
            request.put("seq_id", "test001");
            request.put("task_id", "20001");
            request.put("image", Arrays.asList("yourImageUrl"));
            request.put("batch_size", 1);
            request.put("style", 0);
            request.put("callback_url", "yourCallbackUrl");
            // ########JsonNode
            String requestString = JSONUtil.toJsonStr(request);
```

```
            // ##HTTP##
            HttpResponse response =
                    HttpRequest.post(url)
                            .headerMap(headers, true)
                            .body(requestString)
                            .execute();
            // ######
            System.out.println(response.body());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

Python示例

```
import requests
import json
import logging

# ####
logging.basicConfig(level=logging.INFO, format=\'%(asctime)s - %(levelname)s -
 %(message)s\')
logger = logging.getLogger(__name__)

def send_post_request_enhanced():

    try:
        url = "######"

        headers = {
            "Content-Type": "application/json",
            "X-APP-ID": "xxxxx",
            "Authorization": "xxxx",
            # "Device-Uuid": "yourDeviceUuid"  # ##
        }

        # #####
        request_body = {
            "seq_id": "test001",
            "task_id": "20001",
            "image": ["http://yourImageUrl"],
            "batch_size": 1,
            "style": 0,
            "callback_url": "yourCallbackUrl"
        }

        logger.info(f"####POST###: {url}")
        logger.info(f"###: {headers}")
        logger.info(f"###: {json.dumps(request_body, indent=2,
 ensure_ascii=False)}")

        # ##POST##
        response = requests.post(
            url=url,
            headers=headers,
            json=request_body,
            timeout=30
        )

        logger.info(f"#####: {response.status_code}")

        # ####
        if response.status_code == 200:
            try:
                response_data = response.json()
```

```
                logger.info("POST####")

                # ######code##
                code = response_data.get("code")
                message = response_data.get("message")
                logger.info(f"code: {code}")
                logger.info(f"message: {message}")

            except json.JSONDecodeError:
                logger.error("#######JSON##")
                logger.error(f"####: {response.text}")
                return response.text
        else:
            logger.error(f"POST########: {response.status_code}")
            logger.error(f"####: {response.text}")
            return None

    except requests.exceptions.Timeout:
        logger.error("####################")
    except requests.exceptions.ConnectionError:
        logger.error("########URL############")
    except requests.exceptions.HTTPError as e:
        logger.error(f"HTTP##: {e}")
    except Exception as e:
        logger.error(f"######: {e}")

# #######
if __name__ == "__main__":
    result = send_post_request_enhanced()
    if result:
        print("######")
```