目　录

- 07-accounting-api.md
- 06-billing-api.md
- 05-order-api.md
- 04-transaction-api.md
- 03-wallet-api.md
- 02-auth-api.md
- 01-user-api.md
- 00-overview.md
- 00e-baofoo-sms-api.md
- 00d-preparation-checklist.md
- 00c-tech-stack.md
- 00b-portal-guide.md
- README.md

V2

- 规则
  - Python项目规则
- Ai-hub数据库表结构
  - 数据库表

ai-hub

ai-hub-chart

文本视频生成

AI平台与BIII对账

openAI-Anthropic

AI平台使用使用资金

工作量-企业角色

OAUTH-接入

AI平台与BIII对账（4-13）

添加销售角色（4-13）

企业级邮箱接入（4-13）

Ai-model-网址配置（4-13）

AI-admin-替换主题（04-14）
工作量统计
需要控制的菜单权限
权限方案2
权限方案3
日志错误分析
Ai-hub数据库表结构
Token支付流程安全问题
支付的核心流程
根据模型阶梯扣费SQL
统计-模型错误占比

项目管理

环境管理

各平台账号管理

云平台账号信息
云平台 访问地址 账号 密码
测试环境 http://10.254.240.211 17600000000 Cloud@admin0001
测试环境MYSQL 云平台独立数据库连接：
地址： 10.0.20.108
账号： tx_mandao_cloud_rw
密码： o9J%3Pk5DGOMuHgL$nt2
18888888888 Cloud@test0001
渗透环境 http://10.254.243.68 17600000000 Cloud@admin0001
18888888888 Cloud@test0001
准生产环境 http://10.254.216.130 17600000000
Cloud@admin0001
18888888888 Cloud@test0001
生产环境
华东-上海
https://cloud.mandao.com
http://10.6.122.51:30080

宝信：
https://101.89.84.35

南汇2：
https://222.73.110.145 17600000000 Cloud@admin0001
18888888888 Cloud@test0001
阿里云环境 https://yun.mandao.com

http://47.102.129.66:30001/
http://jsstep.mandao.com/#!/home
http://cloud.jsstep.com/#!/home 17600000000 CLOUD888
18888888888 CLOUD666

镜像仓库访问账号密码
云平台 访问地址 账号 密码 备注
测试环境 dockerhub.baofu.com
dockerhub.test.com admin Harbor@Superme001 备注：hosts
10.254.216.17 dockerhub.baofu.com
10.254.216.17 dockerhub.baofu.com
对外仓库 dockerhub.mandao.com admin bx-mandao@64
宝信仓库 https://172.20.30.74:443 admin prod-Dockerhub@64

数据库账号
云平台 访问地址 账号 密码 备注
测试环境 10.0.20.108 3306 tx_mandao_cloud_rw
o9J%3Pk5DGOMuHgL$nt2
参透环境 10.254.200.110 3306 root o9J%3Pk5DGOMuHgL$nt2
生产 172.32.6.20 3306 arch_weiwei cUzl6L49EJxdhOMV

帮助中心
云平台 访问地址 账号 密码 备注
华东-上海 https://cloud-help.mandao.com/login admin mdy64@help

vSphere账号 vSphere
Esxi8.0 client 访问地址
客户端地址： 10.0.26.10
root/baofoo@64

镜像仓库

生产

- 请求地址

  dockerhub-ha.mandao.com

生产

地址

高可用：
dockerhub-ha.mandao.com
高可用镜像仓库以172.20.30.74为主，其他几台备份
172.20.30.74
172.20.30.101
192.168.222.1
192.168.222.2
宝信：
172.20.30.74 目前使用的宝信镜像仓库
172.20.30.101 目前作为172.20.30.74备份仓库，从172.20.30.74同步镜像
172.20.30.102 公网对外宝信镜像仓库dockerhub.mandao.com与172.20.30.103互备
172.20.30.103 公网对外宝信镜像仓库dockerhub.mandao.com与172.20.30.102互备
南汇：
192.168.222.1 目前作为172.20.30.74备份仓库，从172.20.30.101同步镜像（定期备份）
192.168.222.2 目前作为172.20.30.74备份仓库，从172.20.30.101同步镜像
深圳：
dockerhub-hk.mandao.com 目前作为[华南-深圳]环境使用的仓库
通过helm安装

测试

dockerhub.baofu.com
通过helm安装，数据保存在10.6.122.48上的/data/harbor2

客户管理

爱以德

爱以德项目信息
https://xcn7gxx6wv3z.feishu.cn/wiki/H2vUwV79wiJsS8kt5M7cEhEPnDh
openstack云平台 访问地址 账号 密码 备注
vip 10.10.11.246 mstack-environment.com
server01 10.10.11.247 root 1234
server02 10.10.11.248 root 1234
server03 10.10.11.249 root 1234
ceph https://10.10.11.247:8443 admin M..
openstack 10.10.11.246 admin
PCabg9PhDjTj5q37EORKeKnin1lRqIxNgGXxXOf9

k8s云平台 访问地址 账号 密码 备注
vip 10.10.11.250 dockerhub.zhunyintech.com
node01 10.10.11.251 root 1234
tx252 10.10.11.252 root 1234
tx253 10.10.11.253 root 1234
ceph http://10.10.11.250 admin mandaoyun

香港项目
香港生产堡垒机 http://172.20.192.67/ui/#/profile/improvement
xu_zhang@baofu.com/Baofu*zhang@5321*
vip 172.20.196.100 openstack-cluster-baofoo.com
*node01-17 172.20.196.17 deploy root*pass: xinyan
node02-18 172.20.196.18 root_pass: xinyan
node03-19 172.20.196.19 root_pass: xinyan

测试环境33
vip 10.6.122.60 testing-environment-60.com
node33 10.6.122.33
node23 10.6.122.23
node27 10.0.19.27

爱以德VPN连接信息如下： http://10.10.11.253:30088/
地址： https://61.169.212.66:4430/portal/#!/service admin 业务
地址： 科技网vpn https://61.169.212.66:4430
账号密码： mandaoyun2 / Mandao@64 mandaoyun@64/baofoo@64
vlan80 10.10.8.251-253 3ip https://gvpn.aiyidegroup.com:4430
mandaoyun
vlan90 10.10.9.251-253 3ip mandaoyun baofoo@64
漫道云访问地址： vlan100 10.10.10.251-253 3ip mdy@123
https://paascloud.zhunyintech.com vlan110 10.10.11.249-253 5ip
admin
Superme@01 https://10.10.20.1/ tx/tx
http://10.10.11.250:30080/
https://10.10.20.2/ admin
知识库 https://10.10.20.3/ Superme@01
http://10.10.11.253:30088/ admin
admin Password@_
mandaoyun@64

```
    grafana    admin    baofoo@64
```

http://10.10.11.250:30080/
admin
Superme@01

费卡华瑞

费卡华瑞
云平台使用的域名：
cloud.fresenius-kabi.com.cn 10.130.123.226
Demo：
18888888881
loongtest@demo.com/ Cloud666
Admin：
17600000001
loongad@demo.com / Cloud888
镜像仓库：
dockerhub.fresenius-kabi.com.cn 10.130.123.226
admin Harbor123456
http://10.1.60.16:31080/users/sign_in
root Cloud888
jenkins Cloud888
平台账号SVC外部址
vip: 10.130.123.226
rabbitmq: 10.130.123.227
grafana: 10.130.123.228
夜莺: 10.130.123.229
elk: 10.130.123.230
操作文档: 10.130.123.231
nginx: 10.130.123.232 10.233.164.103
网络配置
1.Bond0
DMZ网络: 10.233.164.100-115/24 GW:254 VLAN 164

2.Bond1

Ceph&K8S网络： 10.130.123.223-237/24 GW:254 VLAN 10

3.Bond2

容器业务网络1：Bond2.11 10.156.153.30-45/24 GW:254 VLAN 11

容器业务网络2：Bond2.60 10.233.156.80-95/24 GW:254 VLAN 60

备份主机：

172.18.5.15

root/Step@2023

项目上线流程

容器云平台镜像版本统一与升级操作手册

一、前置说明

（一）操作目标

1. 统一容器云平台（http://10.254.240.211/#!/home）中指定镜像的版本并完成编译锁定；
2. 修改 KubeBiz 平台（https://www.kubebiz.com/）对应镜像版本配置；
3. 后台服务器（10.0.19.238）新增升级所需内容；
4. 同步更新平台安装文档（https://cloud-help.mandao.com/docs/cloud/install-core）。

（二）前置准备

1. 确保已获取各平台登录账号及权限：

- 容器云平台：管理员账号（admin/Superme@01）或测试账号（test/Cloud@888）；
- KubeBiz 平台：对应操作权限账号；
- 后台服务器：root 账号（用户名：root，密码：baofoo@64）。

2. 操作环境要求：

- 容器云平台操作需在已完成 NFS 存储安装的环境中执行；
- 所有服务器操作需确保网络通畅，能够正常拉取镜像及脚本文件；
- 后台服务器操作需在 node01 服务器上执行（统一操作环境）。

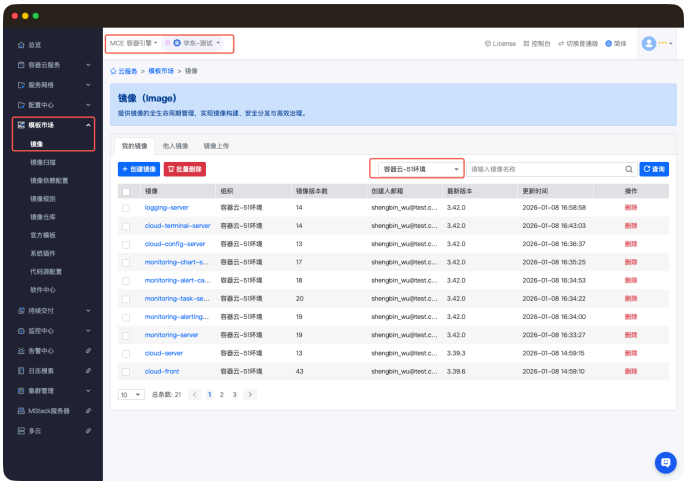
二、容器云平台操作（http://10.254.240.211/#!/home）

（一）登录平台

1. 打开浏览器，输入平台地址：http://10.254.240.211/#!/home ；
2. 在登录页面输入账号密码（管理员账号：admin/Superme@01 或测试账号：test/Cloud@888）；
3. 点击 “登录” 按钮，进入容器云平台首页。

（二）选择镜像信息

1. 登录成功后，在平台顶部导航栏依次点击【云服务】→【模板市场】→【镜像】，进入镜像管理页面；

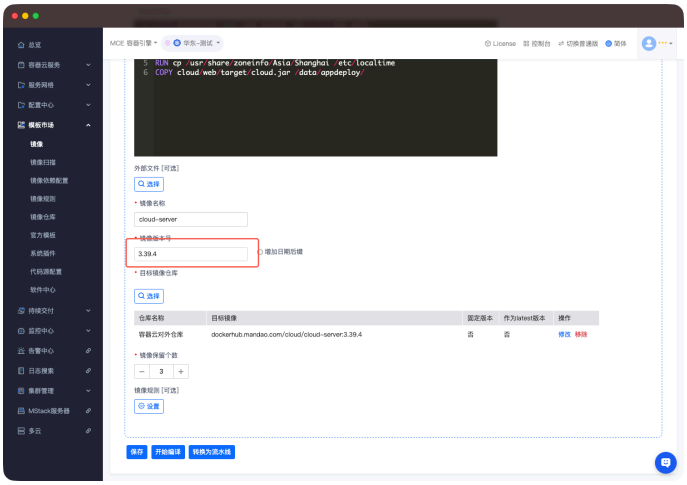


2. 页面将展示容器云 - 51 环境下的所有镜像列表，包含镜像名称、组织、镜像版本数、创建人邮箱、最新版本、更新时间及操作列（示例镜像包括 logging-server、cloud-terminal-server、cloud-config-server 等）；

3. 确认需统一版本的目标镜像范围（建议全量选择列表中所有镜像），核对当前各镜像的版本信息（当前部分镜像最新版本为 3.42.0，部分为 3.39.x 系列）。

（三）统一编译相同版本

1. 选择首个需编译的镜像（如 cloud-server），点击该镜像操作列的相关配置入口，进入镜像编译配置页面；



2. 在编译配置页面按以下参数设置：

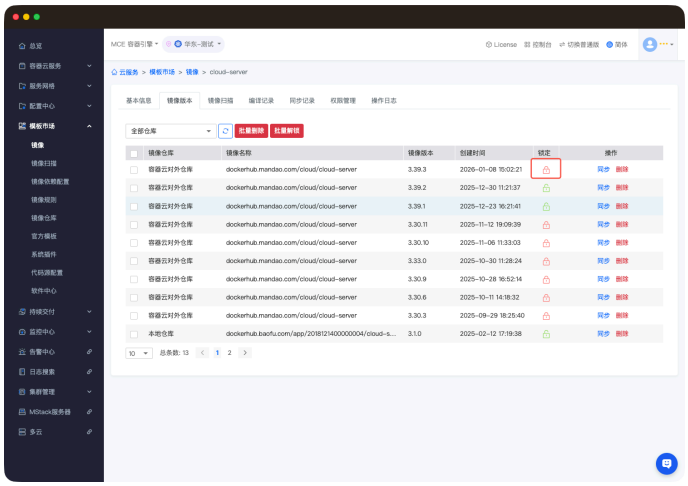
- 镜像名称：确认目标镜像名称（如 cloud-server）；
- 镜像版本：输入统一目标版本号（建议与现有最新版本 3.42.0 保持一致）；
- 目标镜像仓库：选择 “容器云对外仓库”，仓库地址自动填充为 dockerhub.mandao.com/cloud/[镜像名称]；
- 版本设置：取消 “固定版本” 和 “作为 latest 版本” 勾选（如需设置为 latest 可根据需求调整）；
- 镜像保留个数：设置为 3（可根据实际需求调整保留的历史版本数量）；
- 外部文件（可选）：无需添加，保持默认配置；

- 镜像规则（可选）：勾选 “增加日期后缀”（如需追溯编译时间，无需则取消）；
- 编译命令：确认默认编译命令（如 RUN cp /usr/share/zoneinfo/Asia/Shanghai/etc/localtime COPY cloud/web/target/cloud.jar/data/appdeploy），无需修改则保持默认。

3. 配置完成后，点击页面底部 “保存” 按钮，再点击 “开始编译”，启动镜像编译流程；
4. 重复步骤 1-3，依次对镜像列表中所有镜像执行相同配置的编译操作，确保所有镜像均统一编译为目标版本（3.42.0）；
5. 编译过程中可通过【持续交付】→【编译记录】查看各镜像编译进度，若编译失败，可查看日志排查问题（常见问题为网络中断、镜像依赖缺失，需确保网络通畅并补充依赖）。

（四）编译完成后锁定版本

1. 所有镜像编译完成后，返回【镜像】→【镜像版本】页面（或通过镜像名称进入单个镜像的版本管理页面）；



2. 版本列表中将显示各镜像已编译完成的目标版本（3.42.0），找到对应版本行，点击操作列的 “锁定” 按钮；

3. 弹出锁定确认弹窗，点击 “确认” 完成版本锁定；

4. 锁定完成后，版本列表中 “锁定” 列将显示锁定状态，锁定后的版本无法被删除或覆盖，确保版本稳定性；

5. 批量操作说明：若支持批量锁定，可勾选所有已编译完成目标版本的镜像，点击页面顶部 “批量解锁” 旁的 “批量锁定” 按钮，一次性完成所有版本锁定（若平台无批量锁定功能，则按单个版本锁定操作）。

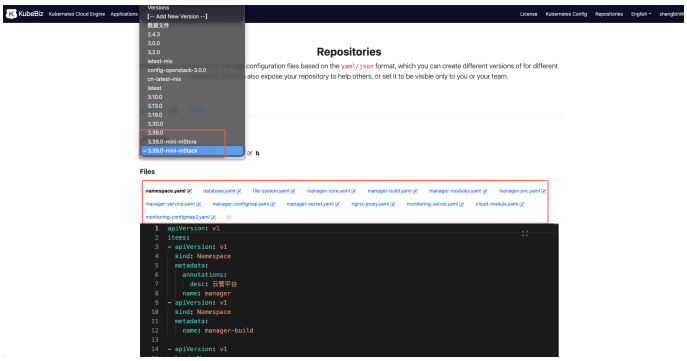
三、KubeBiz 平台操作（https://www.kubebiz.com/）

（一）登录平台

1. 打开浏览器，输入平台地址：https://www.kubebiz.com/ ；
2. 输入已授权的账号密码，点击 “登录” 进入 KubeBiz 平台首页。

（二）修改对应镜像版本

1. 在平台顶部导航栏点击【Kubernetes Config Repositories】，进入配置仓库管理页面；



2. 找到对应容器云平台的配置仓库（命名通常包含 “manager” 或 “cloud” 关键字），点击仓库名称进入文件管理页面；
3. 页面将显示仓库下所有配置文件（如 namespace.yaml、database.yaml、manager-core.yaml、cloud-module.yaml、monitoring-server.yaml 等），根据镜像类型修改对应配置文件：
    - 对于 cloud-server 相关配置：打开 cloud-module.yaml，找到 “image” 字段，将版本号修改为目标版本（3.42.0），示例：dockerhub.mandao.com/cloud/cloud-server:3.42.0；

- 对于监控类镜像（如 monitoring-server、monitoring-alert-callback-server 等）：打开 monitoring-server.yaml 或对应监控模块配置文件，将镜像版本统一修改为 3.42.0；
- 其他镜像（如 logging-server、cloud-terminal-server 等）：在对应的配置文件（如 manager-core.yaml、service.yaml 等）中找到对应镜像地址，修改版本号为 3.42.0。

4. 版本补充：在配置文件顶部 “Versions” 下拉框中，点击【– Add New Version –】，输入新版本名称（如 3.42.0），关联修改后的配置文件，确保新版本配置生效；
5. 若需保留历史版本，可在 “Versions” 中保留原版本（如

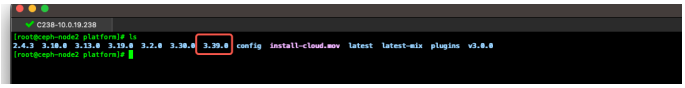
3.39.0、3.30.0 等），新增 3.42.0 版本作为当前生效版本；

6. 修改完成后，点击页面底部 “Save” 按钮保存配置，确保配置同步至容器云平台。

四、后台服务器操作（10.0.19.238）

（一）登录后台管理服务器

1. 打开终端工具（如 Xshell、SecureCRT 等），新建 SSH 连接；



2. 连接参数配置：
    - 主机地址：10.0.19.238；
    - 端口号：22（默认 SSH 端口）；
    - 用户名：root；
    - 密码：baofoo@64；

3. 输入参数后点击 “连接”，验证密码通过后，成功登录服务器终端。

（二）进入目标目录

4. 在终端中执行以下命令，切换至目标目录：

```
cd /data/nfs/manager-pvc-cloud-help-center-uploads-pvc-1724794a-e88f-4dac-bfad-2fecf9ee26e9/cloud/install/platform
```

5. 执行以下命令，查看当前目录下已有的版本文件夹及文件：

```
ls -l
```

6. 确认目录下已存在的版本（如 2.4.3、3.10.0、3.39.0 等），确保目录路径正确。

（三）新增相应内容

7. 执行以下命令，创建目标版本（3.42.0）的文件夹：

```
mkdir -p 3.42.0
```

8. 进入新增的 3.42.0 文件夹：

```
cd 3.42.0
```

9. 下载或复制对应版本的安装脚本至该文件夹，执行以下命令获取完整版、存储化最小化、虚拟化最小化安装脚本（与 3.39.0 版本脚本结构一致，版本号替换为 3.42.0）：

```
# 完整版安装脚本
wget -qO- https://cloud-help.mandao.com/uploads/cloud/install/platform/3.42.0/install.sh -O install.sh
# 存储化最小化安装脚本
wget -qO- https://cloud-help.mandao.com/uploads/cloud/install/platform/3.42.0/install-mini-mStore.sh -O install-mini-mStore.sh
# 虚拟化最小化安装脚本
wget -qO- https://cloud-help.mandao.com/uploads/cloud/install/platform/3.42.0/install-mini-mS
```

```
tack.sh -O install-mini-mStack.
sh
# 插件配置脚本
wget -qO- https://cloud-help.ma
ndao.com/uploads/cloud/install/
platform/plugins/download-plugi
n.sh -O download-plugin.sh
```

4. 若无法通过 wget 下载，可将 3.39.0 版本的脚本复制至 3.42.0 文件夹，再修改脚本内的版本号为 3.42.0（需确保脚本内所有涉及版本的路径、参数均同步更新）：

```
# 复制 3.39.0 版本脚本至 3.42.0
文件夹
cp ../3.39.0/*.sh ./
# 修改脚本内版本号（以 install.
sh 为例）
sed -i 's/3.39.0/3.42.0/g' inst
all.sh
sed -i 's/3.39.0/3.42.0/g' inst
all-mini-mStore.sh
sed -i 's/3.39.0/3.42.0/g' inst
all-mini-mStack.sh
```

5. 执行以下命令，添加脚本执行权限：

```
chmod +x *.sh
```

6. 确认新增内容：执行 ls -l 查看 3.42.0 文件夹下是否包含上述 4 个脚本文件，且权限为 “-rwxr-xr-x”（可执行权限）。

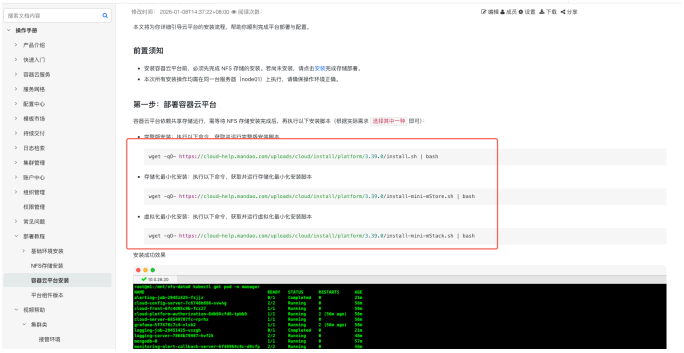
五、更新升级内容（https://cloud-help.mandao.com/docs/cloud/install-core）

（一）访问安装文档页面

1. 打开浏览器，输入文档地址：https://cloud-help.mandao.com/docs/cloud/install-core ；
2. 若文档需要登录，使用容器云平台管理员账号（admin/Superme@01）登录后进入文档页面。

（二）编辑文档更新升级内容

1. 点击文档页面顶部 “编辑” 按钮（通常位于页面右上角，需编辑权限），进入文档编辑模式；



2. 按以下内容更新文档：
    - 文档标题补充：在原标题 “容器云平台安装” 后添加 “（3.42.0 版本升级）”，明确版本信息；
    - 前置须知更新：补充 “本次升级需将所有镜像统一编译为 3.42.0 版本，且已完成镜像版本锁定，确保升级一致性”；
    - 安装脚本更新：将 “第一步：部署容器云平台” 中的 3.39.0

版本脚本路径替换为 3.42.0 版本，更新后脚本如下：

```
# 完整版安装（3.42.0 版本）
wget -qO- https://cloud-help.mandao.com/uploads/cloud/install/platform/3.42.0/install.sh | bash
# 存储化最小化安装（3.42.0 版本）
wget -qO- https://cloud-help.mandao.com/uploads/cloud/install/platform/3.42.0/install-mini-mStore.sh | bash
# 虚拟化最小化安装（3.42.0 版本）
wget -qO- https://cloud-help.mandao.com/uploads/cloud/install/platform/3.42.0/install-mini-mStack.sh | bash
```

- 安装成功效果更新：替换原有的 pod 运行状态示例，补充 3.42.0 版本镜像对应的 pod

信息（如 cloud-server-68549787fc-rprhz（3.42.0）、monitoring-server-xxx（3.42.0）等）；

- 升级说明补充：在文档末尾新增 “3.42.0 版本升级说明”，内容如下：
- 本次升级统一了容器云 - 51 环境下所有镜像版本至 3.42.0，包含 logging-server、cloud-terminal-server、cloud-config-server 等核心镜像；
- 升级前需确保所有镜像已完成编译并锁定，避免版本冲突；
- 升级过程中若出现 pod 启动失败，可通过 kubectl logs [pod名称] -n manager 查看日志，排查镜像拉取失败或配置错误问题；

- 升级完成后，需登录 KubeBiz 平台确认配置文件版本同步更新，确保集群运行配置与镜像版本一致。

3. 编辑完成后，点击页面底部 “保存” 按钮，提交文档更新；
4. 保存后，刷新文档页面，验证更新内容是否生效，确保脚本路径、版本号、升级说明均正确显示。

六、操作验证与注意事项

（一）操作验证

1. 容器云平台验证：
    - 进入【镜像】→【镜像版本】，确认所有镜像目标版本（3.42.0）已锁定；
    - 执行命令 kubectl get pod -n manager，查看所有 pod 对应的镜像版本是否为 3.42.0，且状态为 “Running”。

2. KubeBiz 平台验证：

- 进入配置文件详情页，确认所有镜像地址版本已更新为 3.42.0；
- 查看集群运行状态，确保无配置冲突导致的服务异常。

3. 后台服务器验证：

- 进入 /data/nfs/manager-pvc-cloud-help-center-uploads-pvc-1724794a-e88f-4dac-bfad-2fecf9ee26e9/cloud/install/platform/3.42.0 目录，执行 ./install.sh –version（若脚本支持版本查询），验证脚本版本为 3.42.0。

4. 文档验证：

- 访问文档页面，确认所有脚本路径、版本号、升级说明已更新正确。

（二）注意事项

1. 编译镜像时，需确保网络通畅，镜像拉取可能耗时较长（镜像包体积较大），避免中途中断操作；

2. 版本锁定后，若需修改版本，需先解锁再操作，解锁前需确认无服务依赖该版本；

3. 后台服务器脚本修改时，需确保所有涉及版本号的参数均同步更新，避免脚本执行失败；

4. 升级过程中若出现服务异常，可通过【监控中心】→【告警中心】、【日志搜索】排查问题，必要时回滚至历史版本（3.39.0）；

5. 所有操作需在非业务高峰期执行，避免影响线上服务运行。

AI-算力平台

关于登录的改造（2026-07-15）

关于登录接口认证机制

登录流程（所有入口都汇聚到 setupLogin）：

user.go

```
func setupLogin(user *model.User, c *gin.Context) {
    session := sessions.Default(c)
    session.Set("id", user.Id)
    ...
    err := session.Save()
```

所有登录入口（Login、oauth.go:127、twofa.go:486、passkey.go:325、telegram.go:98、wechat.go:122）都调用 setupLogin，把用户信息写进 cookie session，不返回任何 token。
Session store 是 cookie-based（main.go:151），密钥 common.SessionSecret（未配置时是每次启动随机的 UUID）。

** 鉴权流程 authHelper（middleware/auth.go:33）：**

先读 session，取 username/role/id/status。
session 没有时，读 Authorization header，当作数据库里存的随机 access_token（model.ValidateAccessToken，非 JWT，只是 29-32 位随机串）。
无论哪种方式，都强制要求 Ai-Hub-User header 且必须等于 id（auth.go:96）。
status/role/group 全部来自 session（或 DB token 查出的 user）。

1. 新增 JWT 工具（新文件，如 common/jwt.go 或 service/jwt.go）

golang-jwt/jwt/v5 已在 go.mod，无需引依赖。
提供 GenerateJWT(user) 和 ParseJWT(tokenString)；claims 建议放 id/username/role/status/group/org_id/is_org_admin/is_reseller + exp/iat。
用 HS256，密钥用 common.SessionSecret 或新增一个专门的 JWT_SECRET。

2. setupLogin（controller/user.go:97）

生成 JWT，放进返回 JSON 的 data 里（如 “access_token”: jwt）。
因为所有登录入口都走这里，一处改动即可覆盖全部登录方式。
session 可保留（渐进迁移）也可移除。

3. authHelper（middleware/auth.go:33）

在 username == nil 分支，把 Authorization: Bearer 解析出来（区别于旧的 ValidateAccessToken）。
从 claims 填充 username/role/id/status/group/org_id/is_org_admin。
决定是否保留旧的 DB access_token 兜底（ValidateAccessToken）以做兼容。

4. 关联但容易漏的地方

OptionalUserAuth / TryUserAuth / TokenOrUserAuth（同文件）也直接读 session，若这些路由被前端使用，同样要支持 JWT。
Logout（user.go:143）目前只清 session，JWT 无状态无法真正失效，需要重新设计。
前端：登录后要保存 token 并在每个请求带上 Authorization: Bearer（以及现有的 Ai-Hub-User header）。

风险点
Ai-Hub-User header 强绑定（auth.go:96）：现在即使有 session 也强制要求这个 header 等于 id。改 JWT 后 id 已在 token 里，这个校验要么保留（前端必须继续发）、要么去掉，否则直接 401。这是最容易踩的坑。

Authorization header 语义冲突：该 header 同时被 API key（TokenAuth，sk- 令牌）和用户 access_token 使用。JWT 进来后要能和旧的 DB access_token、sk- key 区分开（靠 JWT 的三段 . 结构或前缀判断），否则老接口/老客户端会失效。

无状态导致的信息陈旧 / 无法吊销：

JWT 里的 status/role/group/quota 是签发时的快照。用户被封禁、改角色、改分组后，token 到期前依然有效。建议 authHelper 里用 model.GetUserCache(id) 重新校验 status（现在 session 版也有类似问题，但 session 可被 clear，JWT 不行）。
Logout 无法让已签发 JWT 立即失效，需要短过期 + refresh token，或维护黑名单/version 号。
密钥管理：SessionSecret 未设 SESSION_SECRET 时每次启动是随机 UUID（common/constants.go:36）→ 重启后所有 JWT 失效；多实例部署下各节点密钥不一致会导致跨节点校验失败。上线前必须固定 SESSION_SECRET（或独立 JWT 密钥）。

2FA / passkey 中间态依赖 session：Login 在需要 2FA 时把 pending_user_id 写进 session（user.go:63），/login/2fa 再读出来。若彻底移除 session，这段中间态也要改（发一个临时 token）。passkey/secure_verification 同理仍用 session。

前端安全权衡：cookie session 有 HttpOnly + SameSite=Strict（main.go），天然抗 XSS/CSRF。JWT 存 localStorage 易受 XSS 窃

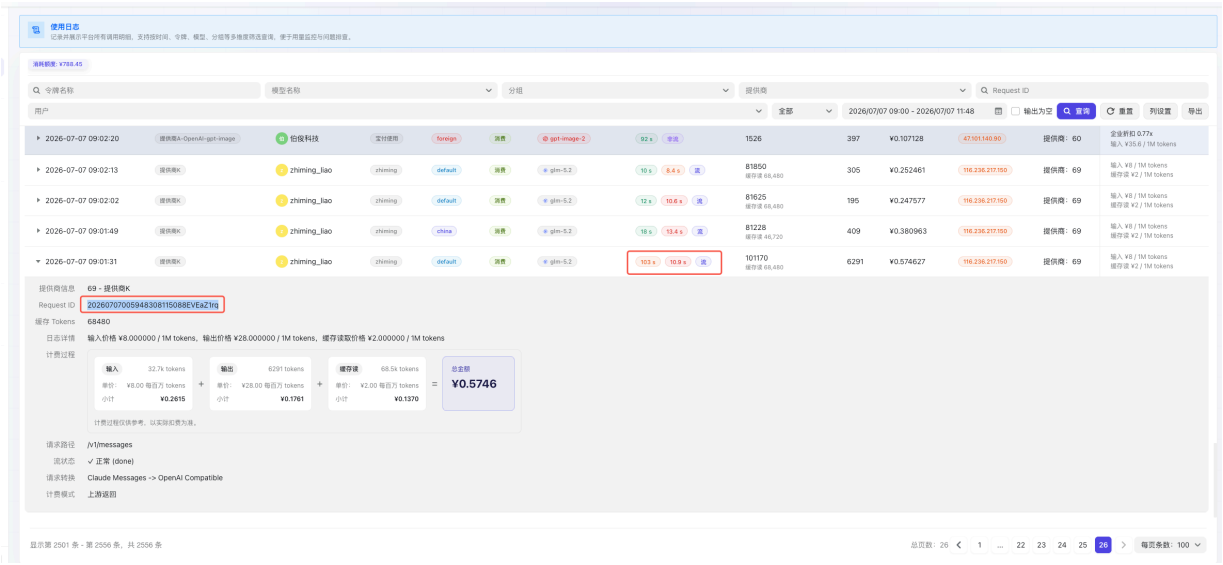
取；存 cookie 又要处理 CSRF。需要明确存储方案。

过期与刷新策略未定：需要确定 access token 有效期、是否引入 refresh token、前端如何续期，否则用户体验（频繁掉线）或安全性会出问题。

关于平台卡顿分析

日志

正常请求 2026070700594830811508 8EVEaZ1rq




```
[root@iZuf6dth8lkakv93dngjacZ ai-hub-platform-backend]# grep "2026070700594830811508 8EVEaZ1rq" ai-hub-api-20260707.log
[INFO]2026-07-07 08:59:48|2026070700594830811508 8EVEaZ1rq|888889150|GetDiscountRatio: matched model discount, user_id=888889150, org_id=24, discount_id=10, discount_type=2, model=glm-5.2, ratio=1.000000
[INFO]2026-07-07 08:59:48|2026070700594830811508 8EVEaZ1rq|888889150|用户 888889150 额度充足，信任且不需要预扣费（funding=wallet）
```

```
[INFO]2026-07-07 08:59:48|20260707005948308115088EVEaZ1rq|0|[relay] POST /v1/messages → upstream https://api.tokenpony.cn/v1/chat/completions (channel #69, type=1, baseURL="https://api.tokenpony.cn", model="glm-5.2", upstreamModel="glm-5.2", relayMode=0)
[INFO]2026-07-07 09:01:31|20260707005948308115088EVEaZ1rq|888889150|stream ended: reason=done
[INFO]2026-07-07 09:01:31|20260707005948308115088EVEaZ1rq|888889150|预扣费后补扣费：¥0.574627（实际消耗：¥0.574627，预扣费：¥0.000000）
[INFO]2026-07-07 09:01:31|20260707005948308115088EVEaZ1rq|888889150|record consume log: userId=888889150, username=zhiming_liao, tokenName=zhiming, group=default, inputTokens=101170, outputTokens=6291, cacheReadTokens=68480, cacheWriteTokens=0
```

请求号 2026070701492473804856 9BbfzUBtF


```
# 第一次

[INFO]2026-07-07 09:49:24|20260707014924738048569BbfzUBtF|888888918|GetDiscountRatio: matched model discount, user_i
```

```
d=888888918, org_id=6, discount_id=11, discount_type=2, model=gpt-image-2, ratio=0.770000
[INFO]2026-07-07 09:49:25|20260707014924738048569BbfzUBtF|888888918|用户 888888918 额度充足，信任且不需要预扣费（funding=wallet）
[INFO]2026-07-07 09:49:25|20260707014924738048569BbfzUBtF|0|[image-relay] model="gpt-image-2", size="2160x3840", n=0 x3dd9a5230600, channelId=#60, apiType=0, baseURL="http://116.236.217.147:30304", requestPath="/v1/images/edits"


# 第二次

[ERR]2026-07-07 10:05:11|20260707014924738048569BbfzUBtF|888888918|do request failed: Post "http://116.236.217.147:30304/v1/images/edits": readfrom tcp 172.22.0.84:58014->116.236.217.147:30304: write tcp 172.22.0.84:58014->116.236.217.147:30304: write: connection reset by peer
[ERR]2026-07-07 10:05:11|20260707014924738048569BbfzUBtF|888888918|channel error (channel #60, status code: 500): upstream error: do request failed
[INFO]2026-07-07 10:05:11|20260707014924738048569BbfzUBtF|888888918|record error log: userId=888888918, channelId=60, modelName=gpt-image-2, tokenName=宝付使用, content=status_code=500, upstream error: do request failed
[INFO]2026-07-07 10:05:11|20260707014924738048569BbfzUBtF|888888918|GetDiscountRatio: matched model discount, user_id=888888918, org_id=6, discount_id=11, discount_type=2, model=gpt-image-2, ratio=0.770000
```

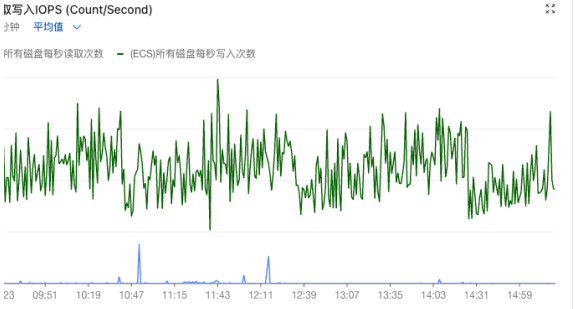
```
[INFO]2026-07-07 10:05:11|20260707014924738048569BbfzUBtF|0|[image-relay] model="gpt-image-2", size="2160x3840", n=0 x3dd9afaeb5e8, channelId=#24, apiType=0, baseURL="https://api.modelverse.cn", requestPath="/v1/images/edits"

# 第三次

[ERR]2026-07-07 10:08:37|20260707014924738048569BbfzUBtF|888888918|do request failed: Post "https://api.modelverse.cn/v1/images/edits": write tcp 172.22.0.84:47316->106.75.185.110:443: write: connection reset by peer
[ERR]2026-07-07 10:08:37|20260707014924738048569BbfzUBtF|888888918|channel error (channel #24, status code: 500): upstream error: do request failed
[INFO]2026-07-07 10:08:37|20260707014924738048569BbfzUBtF|888888918|record error log: userId=888888918, channelId=24, modelName=gpt-image-2, tokenName=宝付使用, content=status_code=500, upstream error: do request failed
[INFO]2026-07-07 10:08:37|20260707014924738048569BbfzUBtF|888888918|GetDiscountRatio: matched model discount, user_id=888888918, org_id=6, discount_id=11, discount_type=2, model=gpt-image-2, ratio=0.770000
[INFO]2026-07-07 10:08:37|20260707014924738048569BbfzUBtF|0|[image-relay] model="gpt-image-2", size="2160x3840", n=0 x3dda1e262d68, channelId=#24, apiType=0, baseURL="https://api.modelverse.cn", requestPath="/v1/images/edits"
```


1 bit/s
1 bit/s
bit/s
09:23
09:52
10:21
10:50
11:19
11:48
12:17
12:46
13:15
13:44
14:13
14:42
15:11


1.907 M bit/s
1.431 M bit/s
976.563 K bit/s
488.281 K bit/s
0 bit/s
09:23
09:53
10:23
10:53
11:23
11:53
12:23
12:53
13:23
13:53
14:23
14:53


取写入IOPS (Count/Second)
分钟 平均值
所有磁盘每秒读取次数
(ECS)所有磁盘每秒写入次数
23
09:51
10:19
10:47
11:15
11:43
12:11
12:39
13:07
13:35
14:03
14:31
14:59


eRDMA流量使用信息 (bit/s)
周期：1分钟 监控值
暂无数据


PPS使用信息 (count/s)
分钟 监控值
暂无数据


同时连接数 (Count)
周期：1分钟 最大值
同时连接数
2026年7月7日 10:34:00
同时连接数 274.000
300.000
250.000
200.000
150.000
100.000
09:23
09:51
10:19
10:47
11:15
11:43
12:11
12:39
13:07
13:35
14:03
14:31
14:59

关于模型的查询接口2026-06-22

小程序定价接口，根据用户身份走不同逻辑

未登录用户（id == 0）

1. 读取「Playground 游客设置」
2. 若开启了游客展示指定用户定价：
    - 加载该指定用户的信息
    - 取该用户所属分组可用的渠道组（usableGroups）
    - 用 usableGroups 过滤定价列表，再过滤掉未配置价格的条目，再过滤掉模型元数据不匹配的条目
    - 以该指定用户的分组身份构建并返回定价响应
3. 否则（普通游客）：
    - 仅过滤未配置价格 + 模型元数据不匹配的条目
    - 以空分组身份返回定价响应

已登录用户（id != 0）

1. 调用 resolvePlaygroundBillingContextWithQuotaCheck 解析 Playground 计费上下文，同时做配额检查（若失败则返回错误）
2. 取计费用户（BillingUserID，可能是本人或其关联的主账户）的用户信息
3. 取该用户可用的渠道组
4. 同样过滤定价（按可用组 → 已配置价格 → 模型元数据匹配）
5. 以该用户的分组身份构建并返回定价响应

——— 改动业务 （新增）如下 ———

```
- 加载该指定用户的信息

获取(usableGroups)
## 添加部分
 - 根据用户自己配置提供商分组
 - 根据用户组织查询提供商分组
 - 用户可以分组

- 取该用户所属分组可用的渠道组（usableGroups）
- 用 usableGroups 过滤定价列表，再过滤掉未配置价格的条目，再过滤掉模型元数据不匹配的条目
- 以该指定用户的分组身份构建并返回定价响应
```

关于成本中心人员-2026-06-16

下面是整理后的流程说明，可以直接放到需求文档或评审说明里。

关于成本中心人员计价流程

现有流程

- 请求进入 middleware/auth.go:305 附近的 Token 鉴权逻辑。
- 请求进入 controller/relay.go:154 附近的转发计费逻辑。
- 系统先根据用户信息、模型、token 数量、提供商分组等信息，获取当前生效的折扣价格。
- 进入channels重试循环：

调整后流程

- 请求进入 middleware/auth.go:305 附近的 Token 鉴权逻辑。
- 在获取 token 用户信息时，额外根据成本中心配置判断当前用户是否为成本人员。
- 将“是否成本人员”的标识写入请求上下文，供后续 relay 计费流程使用。

- 请求进入 controller/relay.go:154 附近的转发流程。
- 进入channels重试循环：
- 如果当前用户是成本人员，则根据本次实际选中的 channel，读取 channel 上绑定的 channel_discount_config_id。
- 根据 channel_discount_config_id 查询渠道折扣配置**。
- 进入channels重试循环：

2026-06-12

第三方 逻辑分析

目标函数：controller/oauth.go:131

这个函数处理的是 OAuth 账号绑定流程，不是 OAuth 登录注册流程。它只会在用户已经登录的情况下执行：主入口 controller/oauth.go:43
会先校
验 state，然后判断 session 里是否已有 username，如果有，就进入 handleOAuthBind。

触发条件

入口逻辑在 HandleOAuth 中：

1. 根据路由参数 provider 获取 OAuth provider。
2. 校验 OAuth state，用于防 CSRF。
3. 如果当前 session 中存在 username，说明用户已登录。
4. 已登录时不走登录/注册，而是调用：

   handleOAuthBind(c, provider)

   所以 handleOAuthBind 的语义是：

   > 当前登录用户把第三方 OAuth 账号绑定到自己的系统账号。

执行流程

1. 检查 Provider 是否启用

如果当前 OAuth provider 被关闭，直接返回错误。

这里虽然主流程后面也会检查 provider enabled，但绑定分支提前 return 了，所以绑定函数里必须自己再检查一次。

2. 用授权码换 Token

从回调 URL 中读取 code，调用 provider 的 ExchangeToken 换取访问令牌。

如果失败，进入统一 OAuth 错误处理：

3. 用 Token 获取 OAuth 用户信息

返回的核心字段是 oauthUser.ProviderUserID，用于标识第三方平台上的用户 ID。

例如 GitHub 当前逻辑中，ProviderUserID 是 GitHub numeric id，同时 Extra[“legacy_id”] 保存旧逻辑使用的 GitHub login。

4. 检查该 OAuth 账号是否已被绑定

位置：controller/oauth.go:153

如果第三方账号 ID 已经存在绑定关系，拒绝绑定。

不同 provider 的检查方式不同：

- 内置 provider：通常查 users 表上的字段，比如 github_id、discord_id、oidc_id。
- 自定义 OAuth provider：查 user_oauth_bindings 表。

5. 检查 legacy_id，避免迁移期重复绑定

这是为了兼容历史数据。

典型场景是 GitHub：

- 老版本可能用 GitHub login 作为绑定 ID。
- 新版本改成 GitHub numeric id。
- 为避免同一个 GitHub 账号在迁移期间被重复绑定，会同时检查新 ID 和旧 ID。

6. 从 Session 获取当前登录用户

这里依赖 session 中存在 id，并且类型必须是 int。

如果查不到用户或数据库错误，返回 common.ApiError。

潜在风险：这里直接做了 id.(int) 类型断言。如果 session 异常、缺失或类型不是 int，会 panic。不过正常路径下，登录时 controller/
user.go:97 会写入 id。

7. 根据 Provider 类型执行绑定

位置：controller/oauth.go:176

这里分两类处理。

自定义 OAuth Provider

自定义 provider 不写 users 表字段，而是写 user_oauth_bindings 表。

底层逻辑在 model/user_oauth_binding.go:108：

1. 检查 provider_id + provider_user_id 是否已经绑定给其他用户。
2. 如果当前用户没有该 provider 的绑定，则创建。
3. 如果当前用户已有该 provider 的绑定，则更新 provider_user_id。

   也就是说，自定义 OAuth 支持“重新绑定同一个 provider 下的另一个第三方账号”。

内置 OAuth Provider

8. 返回绑定成功

整体流程图

```
OAuth callback
     |
     v
HandleOAuth
     |
     v
校验 provider 是否存在
     |
     v
校验 state
     |
     v
session.username 存在？
     |
     +-- 否 --> OAuth 登录/注册流程
     |
     +-- 是 --> handleOAuthBind
                    |
                    v
               provider 是否启用？
                    |
```


v
code 换 token
|
v
token 拉取 OAuth 用户信息
|
v
ProviderUserID 是否已被绑定？
|
v
legacy_id 是否已被绑定？
|
v
读取当前登录用户
|
v
自定义 provider？
|
写 user_oauth_bindings
更新 users 表 provider 字段
返回绑定成功

关键业务规则

- 只有已登录用户才会进入绑定流程。
- 一个 OAuth 第三方账号不能绑定到多个系统用户。
- 自定义 OAuth provider 的绑定关系存储在 user_oauth_bindings。
- 内置 OAuth provider 的绑定关系存储在 users 表对应字段。
- GitHub 等 provider 支持 legacy_id 检查，用于防止历史 ID 迁移期间重复绑定。
- 绑定成功后不会重新登录，也不会刷新 session 用户信息，只返回绑定成功。

需要注意的点

1. session.Get(“id”) 直接类型断言为 int，session 异常时可能 panic。
2. 自定义 OAuth 的 UpdateUserOAuthBinding 支持覆盖当前用户已有绑定，等价于“重新绑定”。
3. 内置 provider 的 user.Update(false) 会更新整个用户模型，具体更新范围取决于 model.User.Update 的实现。
4. provider.IsUserIDTaken 和后续写入不是一个事务，理论上存在并发重复绑定竞争。不过数据库唯一索引或字段唯一约束是否兜

底，需要看具体
provider 对应字段和表结构。

2026-06-02

特殊渠道

```
开始
  ↓
【for 循环】重试次数 ≤ 最大重试次数
  ↓
是否有已锁定的通道（lockedCh）？
  ├── 是 → 使用该锁定通道
  │        ↓
  │       是否是第1次以上重试？
  │        ├── 是 → 执行 SetupContextForSelectedChannel
  │        │        ↓
  │        │       Setup 是否失败？
  │        │        ├── 是 → 封装错误 → break 退出循环
  │        │        └── 否 → 继续
  │        └── 否 → 跳过 Setup
  └── 否 → 调用 getChannel 获取通道
           ↓
          获取通道是否失败？
           ├── 是 → 记录日志 → 封装错误 → break
           └── 否 → 继续
  ↓
addUsedChannel（记录使用通道）
  ↓
获取 bodyStorage
  ↓
获取 body 是否失败？
  ├── 是 → 封装对应错误 → break
```

```
   └── 否 → 重置 Request.Body
   ↓
执行 relay.RelayTaskSubmit 提交任务
   ↓
任务是否成功（taskErr == nil）？
   ├── 是 → break 退出循环（成功结束）
   └── 否 → 继续处理错误
   ↓
是否非本地错误？
   ├── 是 → 执行 processChannelError
   └── 否 → 跳过
   ↓
判断是否需要继续重试（shouldRetryTaskRelay）
   ├── 是 → retry++ → 回到循环开头
   └── 否 → break 退出循环
   ↓
结束
```

逐行逻辑拆解（从上到下顺序判断，命中即终止后续判断）

`if openaiErr == nil { return false }`

没有错误、请求正常返回，不重试。

`if service.ShouldSkipRetryAfterChannelAffinityFailure(c) { return false }`

开启了通道亲和锁定（成功后切换亲和关闭），上一个通道失败后固定不再换渠道，不重试。

`if types.IsChannelError(openaiErr) { return true }`

底层通道故障（网络超时、服务商连接失败、服务宕机等），强制重试。

```
if types.IsSkipRetryError(openaiErr) { return false }
```
业务类不可重试错误（参数错误、模型不存在、权限不足等），不重试。
```
if retryTimes <= 0 { return false }
```
剩余可重试次数用尽，不重试。
```
if _, ok := c.Get("specific_channel_id"); ok { return false }
```
本次请求手动指定了单一固定通道，不允许自动换服务商，不重试。
```
if code >= 200 && code < 300 { return false }
```
HTTP 2xx 成功状态码，业务正常，不重试。
```
if code < 100 || code > 599 { return true }
```
状态码不在 HTTP 标准区间，属于底层协议异常，强制重试。
```
if operation_setting.IsAlwaysSkipRetryCode(openaiErr.GetErrorCode()) { return false }
```
错误码在后台配置的「禁止重试黑名单」，不重试。
```
return operation_setting.ShouldRetryByStatusCode(code)
```
最终依据后台自定义配置：当前 HTTP 状态码是否允许重试。

2025-05

渠道组

“# 渠道分组过滤业务流程说明

> 文档生成时间：2025-05-29
> 最后更新：2025-05-29

概述

本文档详细描述了AI Hub平台的渠道分组过滤业务流程，包括渠道选择逻辑、优先级规则、配置解析顺序以及各层级渠道配置的处理机制。

业务背景

在AI Hub平台中，用户使用AI服务时需要通过渠道进行请求转发。为了满足不同用户、组织和分销商的渠道使用需求，系统实现了多层次的渠道分组过滤机制。

核心概念

1. 渠道组 (Channel Groups)

- 预定义的渠道分组，如 `default`、`premium`、`enterprise` 等
- 每个渠道组包含一组具体的渠道实例

2. 配置字段

- UseChannels: 渠道白名单，逗号分隔的渠道组名称
- ExcludeChannels: 渠道黑名单，逗号分隔的渠道组名称

3. 配置层级

- 组织层级: 组织级别的渠道配置
- 用户层级: 用户个人的渠道配置
- 分销商层级: 分销商相关的渠道配置

业务流程总览

主流程：GetChannelGroupsHandler

详细处理逻辑

1. 缓存读取与降级处理

入口函数: `GetChannelGroupsHandler(userId int, autoChannels []string) []string`

- 输入: 用户ID + 系统自动筛选的基础渠道组
- 第一步: 尝试读取用户缓存信息
- 降级策略: 如果缓存读取失败，直接返回 `autoChannels` ，不做额外过滤
- 目的: 确保系统在异常情况下仍能正常工作

2. 组织链渠道配置解析

函数: `getChannelGroupsOrgHandle(orgId int, autoChannels []string) ([]string, bool)`

处理逻辑：

1. 当前组织检查: 检查当前组织是否配置了 `UseChannels`
2. 递归向上查找: 如果当前组织无配置，递归向上级组织查找
3. 顶级组织处理: 到达顶级组织后，查找组织类型分销商配置
4. 返回值:
    - 过滤后的渠道组列表

- 布尔值表示是否在组织链中找到有效配置

递归查找示例：

```
组织层级: 公司总部 → 事业部 → 部门 → 团队
查找顺序: 团队 → 部门 → 事业部 → 公司总部
```

3. 用户层级配置解析

函数: `getChannelGroupsUserHandle(cache *model.UserBase, autoChannels []string) []string`

处理逻辑：

1. 用户配置优先: 如果用户自身配置了 `UseChannels`，按用户配置过滤
2. 分销商备选: 否则查找个人类型分销商的渠道配置
3. 默认返回: 如果都无配置，返回原始 `autoChannels`

4. 分销商渠道配置

函数: `GetChannelGroupsByResellerByTargetIdAndType(targetType string, targetId int, autoChannels []string) ([]string, bool)`

分销商类型：

- 个人分销商（`ResellerTypePersonal`）：针对单个用户
- 组织分销商（`ResellerTypeOrganization`）：针对整个组织

处理逻辑：

1. 根据目标类型和ID查找对应的分销商
2. 获取分销商父级用户的渠道配置
3. 应用分销商配置到渠道组过滤

5. 渠道组过滤算法

核心函数：`AppendAndRemoveChannelGroups(channels []string, useChannels string, excludeChannels string) []string`

两步过滤策略：

1. 白名单追加（`AppendChannelGroups`）
    - 将 `UseChannels` 中的渠道组追加到基础渠道组
    - 跳过已存在的渠道组，避免重复
    - 忽略空字符串和空白项
2. 黑名单移除（`RemoveChannelGroups`）
    - 从结果中移除 `ExcludeChannels` 指定的渠道组
    - 同样忽略空字符串和空白项

示例：

```
// 输入
channels = [\"default\", \"premium\"]
useChannels = \"enterprise,vip\"
excludeChannels = \"premium\"

// 处理过程
1. 追加白名单: [\"default\", \"premium\", \"enterprise\", \"vip\"]
2. 移除黑名单: [\"default\", \"enterprise\", \"vip\"]
```

配置优先级规则

优先级顺序（从高到低）

1. 组织链配置 ← 最高优先级
2. 用户个人配置
3. 分销商配置 ← 最低优先级

规则说明

- 组织配置优先: 只要组织链中任意级别配置了 `UseChannels`，就以其为准
- 用户配置次之: 仅当组织无配置时，才检查用户个人配置
- 分销商兜底: 前两者都无配置时，才使用分销商配置

- 黑名单覆盖: 黑名单在所有情况下都会生效

异常处理与容错

1. 缓存读取失败

- 场景: 用户缓存信息无法获取
- 处理: 直接返回基础 `autoChannels` ，保障服务可用性

2. 配置解析异常

- 空配置处理: 空字符串或空白项自动忽略
- 无效渠道组: 不存在的渠道组名称会被过滤算法自然排除

3. 递归保护

- 终止条件: 到达顶级组织（ParentId=0）时停止递归
- 循环检测: 依赖数据库约束防止组织链循环引用

性能考虑

1. 缓存优化

- 用户信息缓存减少数据库查询
- 渠道组配置缓存提升过滤性能

2. 递归深度控制

- 组织层级通常不会过深（一般3-5级）
- 递归查找在合理范围内

3. 算法效率

- 使用map进行存在性检查，O(1)时间复杂度
- 避免不必要的数组操作

使用场景示例

场景1：企业统一渠道管理

```
组织配置:
  UseChannels: "enterprise,stable"
  ExcludeChannels: "experimental"
```

```
效果：该组织下所有用户只能使用enterprise和stable渠道组
```

场景2：VIP用户特殊渠道

```
用户配置：
  UseChannels: "vip,premium"
  ExcludeChannels: ""

效果：该用户可以使用vip和premium渠道组，覆盖组织配置
```

场景3：分销商渠道限制

```
分销商配置：
  UseChannels: "partner"
  ExcludeChannels: "internal"

效果：该分销商下的用户/组织只能使用partner渠道组
```

相关配置接口

参考其他文档中的相关接口：

- 组织渠道配置更新接口

- 用户渠道配置更新接口
- 分销商管理接口

注意事项

1. 配置生效时间: 渠道配置修改后需要等待缓存刷新
2. 权限控制: 渠道配置修改需要相应权限
3. 监控告警: 建议监控渠道过滤异常情况
4. 日志记录: 关键操作需要记录审计日志

> 文档版本: v1.0”

V1

14-marketing-api.md

营销模块

营销模块面向平台管理员，提供代金券的全生命周期管理，包括活动创建、券类型定义、批量生成券码、自动/手动发放和数据统计。

> 用户侧的代金券查看与兑换在钱包模块（03）。
> 下单时使用代金券，在订单模块（05）的 `voucher_id` 字段传入。

```
活动（Campaign）
  └──── 券类型（VoucherType）
-- 定义面值/有效期/使用条件
        └──── 券码（VoucherCode）-- 批量生成的实体码
              └──── 发放记录（Distribution）-- 发给哪个用户
```

14.1 营销活动管理

活动列表

GET
`/admin/v1/marketing/campaigns`

查询参数	说明
`status`	`draft` / `active` / `paused` / `ended`
`page`	页码


```
{
  "code": 0,
  "data": {
    "items": [
      {
        "campaign_id": "camp_01HX...",
        "name": "新用户注册礼包",
        "description": "新注册用户即送 20 元代金券",
        "status": "active",
        "trigger": "user_register",
        "start_at": "2024-01-01T00:00:00Z",
        "end_at": "2024-06-30T2
```

```
3:59:59Z",
        "budget_total": "50000.
00",
        "budget_used": "12400.0
0",
        "issued_count": 620,
        "redeemed_count": 310,
        "redeem_rate": "50%"
      }
    ]
  }
}
```

创建活动

POST `/admin/v1/marketing/campaigns`

```
{
  "name": "五一充值优惠",
  "description": "五一期间充值
满 200 元赠 30 元券",
  "trigger": "recharge_threshol
d",
  "start_at": "2024-05-01T00:0
0:00Z",
  "end_at": "2024-05-07T23:59:5
9Z",
```

```
  "budget_total": "10000.00"
}
```

活动详情

GET `/admin/v1/marketing/campaigns/{campaign_id}`

编辑活动

PUT `/admin/v1/marketing/campaigns/{campaign_id}`

上线活动

POST `/admin/v1/marketing/campaigns/{campaign_id}/launch`

暂停活动

POST `/admin/v1/marketing/campaigns/{campaign_id}/pause`

终止活动

POST `/admin/v1/marketing/campaigns/{campaign_id}/end`

活动统计

GET `/admin/v1/marketing/campaigns/{campaign_id}/stats`


```
{
  "code": 0,
  "data": {
    "campaign_id": "camp_01HX...",
    "issued_count": 620,
    "redeemed_count": 310,
    "redeem_rate": "50%",
    "total_discount_given": "6200.00",
    "converted_recharge": "98500.00",
    "roi": "15.9x",
    "daily_trend": [
      { "date": "2024-01-01", "issued": 45, "redeemed": 22 }
```

```
    ]
  }
}
```

14.2 券类型管理

券类型列表

GET `/admin/v1/marketing/voucher-types`

查询参数	说明
`campaign_id`	按活动筛选

```
{
  "code": 0,
  "data": {
    "items": [
      {
        "type_id": "vtype_01HX...",
        "campaign_id": "camp_01HX...",
        "name": "新用户礼包 - 20元券",
```

```
        "voucher_type": "fixed_
amount",
        "amount": "20.00",
        "min_recharge": "0.00",
        "applicable_models":
[],
        "valid_days": 30,
        "per_user_limit": 1,
        "total_limit": 10000,
        "issued_count": 620
      }
    ]
  }
}
```

券类型（voucher_type）枚举：

值	说明
`fixed_amount`	固定金额抵扣
`percentage`	折扣券
`free_tokens`	赠送 Token 额度

值	说明
`model_trial`	指定模型免费试用

创建券类型

POST `/admin/v1/marketing/voucher-types`

```
{
  "campaign_id": "camp_01HX...",
  "name": "新用户礼包 - 20元券",
  "voucher_type": "fixed_amount",
  "amount": "20.00",
  "min_recharge": "0.00",
  "applicable_models": [],
  "valid_days": 30,
  "per_user_limit": 1,
  "total_limit": 10000
}
```

券类型详情

GET `/admin/v1/marketing/voucher-types/{type_id}`

更新券类型

PUT `/admin/v1/marketing/voucher-types/{type_id}`

删除券类型

DELETE `/admin/v1/marketing/voucher-types/{type_id}`

14.3 券码批量生成

批量生成券码

POST `/admin/v1/marketing/voucher-codes/generate`

```
{
  "type_id": "vtype_01HX...",
```

```
  "quantity": 1000,
  "code_prefix": "NEW2024",
  "code_length": 12
}
```

响应（异步任务）：

```
{
  "code": 0,
  "data": {
    "task_id": "gen_task_01H
X...",
    "type_id": "vtype_01HX...",
    "quantity": 1000,
    "status": "processing",
    "estimated_seconds": 5
  }
}
```

查询生成任务状态

GET `/admin/v1/marketing/voucher-codes/generate/{task_id}`

```
{
  "code": 0,
  "data": {
    "task_id": "gen_task_01H
```

```
X...",
    "status": "completed",
    "generated_count": 1000,
    "download_url": "https://oss.example.com/exports/codes_xxx.csv",
    "download_expires_at": "2024-01-01T13:00:00Z"
  }
}
```

券码列表

GET `/admin/v1/marketing/voucher-codes`

查询参数	说明
`campaign_id`	按活动筛选
`type_id`	按券类型筛选
`status`	`unused` / `issued` / `used` / `expired`

查询指定券码详情

GET `/admin/v1/marketing/voucher-codes/{code}`

```
{
  "code": 0,
  "data": {
    "code": "NEW2024XXXXXXXX",
    "type_id": "vtype_01HX...",
    "status": "used",
    "issued_to_user_id": "u_01HX...",
    "issued_at": "2024-01-01T12:00:00Z",
    "used_at": "2024-01-02T10:00:00Z"
  }
}
```

导出券码

GET `/admin/v1/marketing/voucher-codes/export`

14.4 自动发放规则

规则列表

GET `/admin/v1/marketing/distribution-rules`

```
{
  "code": 0,
  "data": {
    "items": [
      {
        "rule_id": "rule_01HX...",
        "name": "新用户注册礼包规则",
        "trigger": "user_register",
        "conditions": {
          "user_type": "personal"
        },
        "voucher_type_id": "vtype_01HX...",
        "status": "enabled",
        "total_issued": 620,
        "created_at": "2024-01-01T00:00:00Z"
      }
    ]
```

```
  }
}
```

触发条件（trigger）枚举：

值	触发时机
`user_register`	新用户注
`first_recharge`	首次充值
`recharge_threshold`	充值满额
`membership_subscribe`	订阅会员
`invitation_success`	邀请好友
`birthday`	用户生日
`inactive_user`	用户 N 天

创建自动发放规则

POST `/admin/v1/marketing/distribution-rules`

```
{
  "name": "新用户注册礼包规则",
  "trigger": "user_register",
```

```
  "conditions": {
    "user_type": "personal"
  },
  "voucher_type_id": "vtype_01HX..."
}
```

更新规则

PUT `/admin/v1/marketing/distribution-rules/{rule_id}`

删除规则

DELETE `/admin/v1/marketing/distribution-rules/{rule_id}`

启用规则

POST `/admin/v1/marketing/distribution-rules/{rule_id}/enable`

禁用规则

POST `/admin/v1/marketing/distribution-rules/{rule_id}/disable`

14.5 手动发放

手动向指定用户发放

POST `/admin/v1/marketing/distribute`

```
{
  "user_ids": ["u_01HX...", "u_02HX..."],
  "voucher_type_id": "vtype_01HX...",
  "reason": "用户投诉补偿"
}
```

响应：

```
{
  "code": 0,
  "data": {
    "success_count": 2,
    "failed_count": 0,
    "failed_users": []
```

```
  }
}
```

批量发放（上传 CSV）

POST `/admin/v1/marketing/distribute/batch`

```
Content-Type: multipart/form-data

file: user_ids.csv
voucher_type_id: vtype_01HX...
reason: 活动补偿
```

CSV 格式（第一列为 user_id）：

```
u_01HX...
u_02HX...
u_03HX...
```

发放记录

GET `/admin/v1/marketing/distributions`

查询参数	说明
`campaign_id`	按活动筛选
`user_id`	按用户筛选
`trigger`	按触发方式筛选（`auto` / `manual`）
`page`	页码


```
{
  "code": 0,
  "data": {
    "items": [
      {
        "dist_id": "dist_01HX...",
        "user_id": "u_01HX...",
        "voucher_type_id": "vtype_01HX...",
        "voucher_code": "NEW2024XXXXXXXX",
        "trigger": "auto",
        "trigger_event": "user_register",
        "issued_at": "2024-01-01T12:00:00Z"
      }
```

```
    ]
  }
}
```

13-message-center-api.md

消息中心

消息中心是用户侧的综合消息收件箱，按类型分类展示，支持搜索、批量操作和 SSE 实时推送。

> 与通知模块的关系：
>
> - 通知模块（12）负责消息的生产与投递（触发 → 生成 → 渠道推送）
> - 消息中心（13）负责消息的展示与管理（收件箱 → 分类阅读 → 搜索归档）

消息分类

type	说明	典型
`announcement`	系统公告	版本 服务

type	说明	典型
`alert`	业务告警	余额用量Key
`security`	安全通知	异地密码
`billing`	账单与支付	充值月度退款
`membership`	会员消息	会员到期权益
`promotion`	活动与优惠	代金限时
`system`	系统任务	数据认证

13.1 消息收件箱

消息列表

GET `/api/v1/messages`

查询参数	类型	说明
`type`	string	消息分类， 支持多选逗号
`is_read`	bool	`true` 已读
`priority`	string	`high` / `no`
`page`	int	页码
`page_size`	int	每页条数


```
{
  "code": 0,
  "data": {
    "items": [
      {
        "msg_id": "msg_01HX...",
        "type": "alert",
        "priority": "high",
        "title": "余额不足提醒",
        "summary": "您的账户余额仅剩 10.00 元",
        "content": "您的账户余额仅剩 10.00 元，距余额耗尽预计还有 2 天，请及时充值。",
        "action": {
```

```
            "label": "立即充值",
            "url": "/wallet/recharge"
        },
        "is_read": false,
        "created_at": "2024-01-01T12:00:00Z",
        "expire_at": null
      }
    ],
    "total": 18,
    "page": 1,
    "page_size": 20
  }
}
```


消息详情（自动标记已读）

GET `/api/v1/messages/{msg_id}`

标记指定消息已读

PUT
`/api/v1/messages/{msg_id}/read`

批量标记已读

PUT `/api/v1/messages/batch-read`

```
{
  "msg_ids": ["msg_01HX...", "msg_02HX..."]
}
```

全部标记已读：

```
{ "all": true }
```

删除消息

DELETE `/api/v1/messages/{msg_id}`

批量删除

DELETE `/api/v1/messages/batch-delete`

```
{ "msg_ids": ["msg_01HX...", "msg_02HX..."] }
```

各分类未读数

GET `/api/v1/messages/unread-count`

```
{
  "code": 0,
  "data": {
    "total": 8,
    "breakdown": {
      "announcement": 1,
      "alert": 3,
      "security": 0,
      "billing": 2,
      "membership": 1,
      "promotion": 1,
      "system": 0
    }
  }
}
```

13.2 消息搜索

GET `/api/v1/messages/search`

查询参数	类型	必填	说明
`q`	string	是	搜索（标

查询参数	类型	必填	说明
			内容
`type`	string	否	限定
`start_date`	string	否	起始
`end_date`	string	否	结束
`page`	int	否	页码

```
{
  "code": 0,
  "data": {
    "items": [...],
    "total": 5,
    "keyword": "余额"
  }
}
```

13.3 实时消息推送（SSE）

GET `/api/v1/messages/stream`

建立 SSE 长连接，服务端实时推送新消息，用于前端顶栏 Badge 实时更

新。

```
GET /api/v1/messages/stream
Authorization: Bearer <token>
Accept: text/event-stream
```

SSE 事件格式：

```
event: new_message
data: {"msg_id":"msg_02HX","type":"billing","title":"充值成功","priority":"normal","is_read":false}

event: unread_count
data: {"total":9,"breakdown":{"billing":3}}

event: ping
data: {"timestamp":1704067200}
```

> `ping` 事件每 30 秒发送一次，用于保活。前端断线后应自动重连（EventSource 原生支持）。

13.4 消息订阅设置

查询订阅设置

GET `/api/v1/messages/subscriptions`

```
{
  "code": 0,
  "data": {
    "subscriptions": [
      { "type": "alert", "in_app": true,  "email": true, "sms": true  },
      { "type": "security", "in_app": true,  "email": true, "sms": true  },
      { "type": "billing", "in_app": true,  "email": true, "sms": false },
      { "type": "membership", "in_app": true,  "email": true, "sms": false },
      { "type": "announcement", "in_app": true,  "email": false, "sms": false },
      { "type": "promotion", "in_app": true,  "email": false, "sms": false },
      { "type": "system", "in_app": true,  "email": fals
```

```
e, "sms": false }
    ]
  }
}
```

更新订阅设置

PUT `/api/v1/messages/subscriptions`

```
{
  "subscriptions": [
    { "type": "alert", "in_ap
p": true, "email": true, "sms":
true }
  ]
}
```

13.5 公告管理（管理员）

公告列表

GET `/admin/v1/messages/announcements`

查询参数	说明
`status`	`draft` / `published` / `unpublished`


```
{
  "code": 0,
  "data": {
    "items": [
      {
        "id": "ann_01HX...",
        "title": "GPT-4o 模型价格调整公告",
        "priority": "high",
        "status": "published",
        "target": {
          "user_type": "all"
        },
        "published_at": "2024-01-01T10:00:00Z",
        "expire_at": "2024-02-01T00:00:00Z",
        "stats": {
          "sent_count": 12800,
          "read_count": 6400,
          "read_rate": "50%"
        }
      }
    ]
```

```
  }
}
```

创建公告

POST `/admin/v1/messages/announcements`

```
{
  "title": "系统维护通知",
  "content": "平台将于 2024-01-10 02:00-04:00 进行例行维护，期间服务不可用。",
  "priority": "high",
  "target": {
    "user_type": "all",
    "membership_levels": []
  },
  "expire_at": "2024-01-11T00:00:00Z"
}
```

编辑公告

PUT `/admin/v1/messages/announcements`

`/{announcement_id}`

删除公告

DELETE `/admin/v1/messages/announcements/{announcement_id}`

发布公告（立即或定时）

POST `/admin/v1/messages/announcements/{announcement_id}/publish`

```
{
  "publish_at": null
}
```

`publish_at` 为 null 表示立即发布，传入未来时间表示定时发布。

撤回公告

POST `/admin/v1/messages/announcements/{announcement_id}/unpublish`

公告触达统计

GET `/admin/v1/messages/announcements/{announcement_id}/stats`

```
{
  "code": 0,
  "data": {
    "sent_count": 12800,
    "read_count": 6400,
    "read_rate": "50%",
    "by_user_type": {
      "personal": { "sent": 10000, "read": 5200 },
      "enterprise": { "sent": 2800, "read": 1200 }
    }
  }
}
```

12-notification-api.md

通知模块

通知模块负责平台各类事件触发的消息推送，支持站内信、邮件、短信三种投递渠道，是消息的生产与投递基础设施。

> 用户侧的消息收件箱详见 13-message-center-api.md。

12.1 通知记录（用户）

通知列表

GET `/api/v1/notifications`

查询参数	说明
`is_read`	`true` / `false`
`type`	通知类型
`page`	页码

```
{
  "code": 0,
  "data": {
    "items": [
      {
        "id": "notif_01HX...",
        "type": "balance.low",
        "title": "余额不足提
醒",
        "content": "您的账户余
额仅剩 10.00 元，请及时充值。",
        "is_read": false,
        "channels": ["in_app",
"email"],
        "created_at": "2024-01-
01T12:00:00Z"
      }
    ],
    "unread_count": 3,
    "total": 25
  }
}
```


通知详情

GET `/api/v1/notifications/{notification_id}`

标记已读

PUT `/api/v1/notifications/{notification_id}/read`

全部标记已读

PUT `/api/v1/notifications/read-all`

删除通知

DELETE `/api/v1/notifications/{notification_id}`

未读数量

GET `/api/v1/notifications/unread-count`

```
{
  "code": 0,
  "data": { "unread_count": 3 }
}
```

12.2 通知偏好设置（用户）

查询偏好设置

GET `/api/v1/notifications/preferences`

```
{
  "code": 0,
  "data": {
    "preferences": [
      {
        "event_type": "balance.low",
        "threshold": "20.00",
        "channels": { "in_app": true, "email": true, "sms": false }
      },
      {
        "event_type": "usage.quota_exceeded",
        "threshold_percent": 90,
        "channels": { "in_app":
```

```
true, "email": true, "sms": tru
e }
      },
      {
        "event_type": "membersh
ip.expired",
        "days_before": 7,
        "channels": { "in_app":
true, "email": true, "sms": fal
se }
      },
      {
        "event_type": "apikey.e
xpired",
        "days_before": 3,
        "channels": { "in_app":
true, "email": true, "sms": fal
se }
      }
    ]
  }
}
```

更新偏好设置

PUT `/api/v1/notifications/preference
s`

```
{
  "preferences": [
    {
      "event_type": "balance.low",
      "threshold": "50.00",
      "channels": { "in_app": true, "email": true, "sms": true }
    }
  ]
}
```

12.3 通知发送管理（管理员）

发送全平台公告

POST `/admin/v1/notifications/broadcast`

```
{
  "title": "系统维护通知",
  "content": "平台将于 2024-01-10 02:00-04:00 进行例行维
```

```
护...",
  "channels": ["in_app", "emai
l"],
  "target": {
    "user_type": "all"
  }
}
```

`target.user_type` 枚举：`all` / `personal` / `enterprise`

通知模板列表

GET `/admin/v1/notifications/template
s`

```
{
  "code": 0,
  "data": {
    "items": [
      {
        "template_id": "tpl_bal
ance_low",
        "event_type": "balance.
low",
        "channel": "email",
        "subject": "【余额提
```

```
醒】您的账户余额不足",
        "body": "尊敬的 {{nickname}}，您的账户余额仅剩 {{balance}} 元...",
        "updated_at": "2024-01-01T00:00:00Z"
      }
    ]
  }
}
```

更新通知模板

PUT `/admin/v1/notifications/templates/{template_id}`

```
{
  "subject": "【余额提醒】您的账户余额不足，请及时充值",
  "body": "尊敬的 {{nickname}}，您的账户余额仅剩 {{balance}} 元，预计还可使用 {{days}} 天..."
}
```

发送日志

GET `/admin/v1/notifications/send-logs`

查询参数	说明
`event_type`	事件类型筛选
`channel`	`email` / `sms` / `in_app`
`status`	`success` / `failed`
`start_date`	开始日期


```
{
  "code": 0,
  "data": {
    "items": [
      {
        "log_id": "nlog_01HX...",
        "event_type": "balance.low",
        "channel": "email",
        "user_id": "u_01HX...",
        "recipient": "user@example.com",
        "status": "success",
```

```
        "sent_at": "2024-01-01T
12:00:00Z"
      }
    ],
    "success_rate": "98.5%"
  }
}
```

11-model-api.md

模型模块

模型模块分三层：提供商 → 模型 → 对话，并包含限流规则、请求队列和降级策略。

```
提供商（Provider） ——— 阿里云、OpenAI、Anthropic...
  └——— 模型（Model） ——— gpt-4o、qwen-turbo...
        └——— 对话调用（Chat） ——— /chat/completions
```

11.1 提供商管理（管理员）

提供商列表

GET `/admin/v1/model-providers`

```
{
  "code": 0,
```

```
  "data": {
    "items": [
      {
        "provider_id": "prov_01
HX...",
        "name": "OpenAI",
        "slug": "openai",
        "base_url": "https://ap
i.openai.com/v1",
        "api_key": "sk-****",
        "status": "enabled",
        "model_count": 8,
        "created_at": "2024-01-
01T00:00:00Z"
      }
    ]
  }
}
```

内置提供商 slug：

slug	平台	代表模
`openai`	OpenAI	GPT-4o GPT-4 Turbo
`aliyun`	阿里云百炼	qwen-turbo,

slug	平台	代表模
		qwen-plus
`anthropic`	Anthropic	claude-3-5-sonnet
`google`	Google	gemini-1.5-pro
`baidu`	百度智能云	ernie-4
`zhipu`	智谱 AI	glm-4
`custom`	自定义（兼容 OpenAI 协议）	任意

新增提供商

POST `/admin/v1/model-providers`

```
{
  "name": "阿里云百炼",
  "slug": "aliyun",
  "base_url": "https://dashscope.aliyuncs.com/compatible-mode/
```

```
v1",
  "api_key": "sk-xxxxx"
}
```

提供商详情

GET `/admin/v1/model-providers/{provider_id}`

更新提供商配置

PUT `/admin/v1/model-providers/{provider_id}`

删除提供商

DELETE `/admin/v1/model-providers/{provider_id}`

启用 / 禁用提供商

POST `/admin/v1/model-providers/{provider_id}/enable`

POST `/admin/v1/model-providers/{provider_id}/disable`

连通性测试

POST `/admin/v1/model-providers/{provider_id}/test`

```
{
  "code": 0,
  "data": {
    "status": "success",
    "latency_ms": 320,
    "message": "API Key 有效，连接正常"
  }
}
```

提供商下的模型列表

GET `/admin/v1/model-providers/{provider_id}/models`

提供商调用统计

GET `/admin/v1/model-providers/{provider_id}/stats`

11.2 模型管理（管理员）

模型列表

GET `/admin/v1/models`

查询参数	说明
`provider_id`	按提供商筛选
`type`	`chat` / `embedding` / `image`
`status`	`enabled` / `disabled`


```
{
  "code": 0,
  "data": {
    "items": [
      {
        "model_id": "model_01H
X...",
        "provider_id": "prov_01
HX...",
        "provider_name": "OpenA
I",
        "model_name": "gpt-4o",
        "display_name": "GPT-4
o",
```

```
        "type": "chat",
        "context_window": 12800
0,
        "max_output_tokens": 40
96,
        "status": "enabled",
        "input_price": "0.028",
        "output_price": "0.08
4",
        "currency": "CNY",
        "unit": "per_1k_tokens"
      }
    ]
  }
}
```

添加模型

POST `/admin/v1/models`

```
{
  "provider_id": "prov_01H
X...",
  "model_name": "gpt-4o",
  "display_name": "GPT-4o",
  "type": "chat",
  "context_window": 128000,
```

```
  "max_output_tokens": 4096
}
```

模型详情

GET `/admin/v1/models/{model_id}`

更新模型配置

PUT `/admin/v1/models/{model_id}`

删除模型

DELETE `/admin/v1/models/{model_id}`

启用 / 禁用模型

POST `/admin/v1/models/{model_id}/enable`

POST `/admin/v1/models/{model_id}/disable`

模型调用统计

GET `/admin/v1/models/{model_id}/stats`

11.3 模型对话（用户）

可用模型列表

GET `/api/v1/models`

仅返回已启用的模型（用户视图），隐藏提供商 API Key 等敏感配置。

模型对话

POST `/api/v1/chat/completions`

兼容 OpenAI Chat Completions 格式。

普通请求：

```
{
  "model": "gpt-4o",
  "messages": [
    { "role": "system", "content": "你是一个有帮助的助手" },
    { "role": "user", "content": "你好，帮我写一首诗" }
```

```
  ],
  "temperature": 0.7,
  "max_tokens": 1024,
  "stream": false
}
```

普通响应：

```
{
  "id": "chatcmpl-01HX...",
  "object": "chat.completion",
  "created": 1704067200,
  "model": "gpt-4o",
  "choices": [
    {
      "index": 0,
      "message": { "role": "assistant", "content": "春风轻抚柳..." },
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 32,
    "completion_tokens": 128,
    "total_tokens": 160
  },
  "x-cost": "0.00448",
```

```
  "x-model-used": "gpt-4o"
}
```

流式请求（stream: true）：

```
POST /api/v1/chat/completions
Accept: text/event-stream
```

```
data: {"id":"chatcmpl-01HX","choices":[{"delta":{"role":"assistant"},"index":0}]}

data: {"id":"chatcmpl-01HX","choices":[{"delta":{"content":"春"},"index":0}]}

data: {"id":"chatcmpl-01HX","choices":[{"delta":{"content":"风"},"index":0}]}

data: [DONE]
```

> `x-model-used` 响应头：当触发降级时，此字段标明实际使用的备用模型。

对话历史

GET `/api/v1/chat/history`

查询参数	说明
`model_id`	按模型筛选
`page`	页码

会话详情

GET `/api/v1/chat/history/{session_id}`

删除会话

DELETE `/api/v1/chat/history/{session_id}`

11.4 限流规则管理（管理员）

限流规则列表

GET `/admin/v1/models/rate-limits`

```
{
  "code": 0,
  "data": {
    "items": [
      {
        "rule_id": "rl_001",
        "model_id": "gpt-4o",
        "target_type": "membership_level",
        "target_value": "free",
        "rpm": 10,
        "rpd": 200,
        "tpm": 40000,
        "tpd": 500000,
        "action_on_exceed": "queue"
      }
    ]
  }
}
```


字段	说明
`rpm`	每分钟请求数（Requests P Minute）
`rpd`	每日请求数（Requests P

字段	说明
	Day）
`tpm`	每分钟 Token（Tokens Per Minute）
`tpd`	每日 Token 数（Tokens Per Day）
`action_on_exceed`	超限行为：`q` 进队列 / `rej` 直接拒绝

创建限流规则

POST `/admin/v1/models/rate-limits`

更新限流规则

PUT `/admin/v1/models/rate-limits/{rule_id}`

删除限流规则

DELETE `/admin/v1/models/rate-limits/{rule_id}`

11.5 请求队列管理

用户侧：查询排队状态

GET `/api/v1/chat/queue/status`


```
{
  "code": 0,
  "data": {
    "queued_requests": [
      {
        "request_id": "req_xyz789",
        "model_id": "gpt-4o",
        "queue_position": 5,
        "estimated_wait_seconds": 12,
        "priority": "P2",
        "status": "queued",
        "created_at": "2024-01-01T12:00:00Z"
      }
    ]
```

```
  }
}
```

用户侧：查询指定请求排队位置

GET `/api/v1/chat/queue/{request_id}/position`

用户侧：取消排队请求

DELETE `/api/v1/chat/queue/{request_id}`

用户侧：申请插队

POST `/api/v1/chat/queue/{request_id}/priority`

消耗插队券或扣减余额，将请求临时提升一级优先级。

优先级规则（从高到低）：

优先级	用户类型
PO	企业用户（主账号）

优先级	用户类型
P1	企业成员 / 个人 premium 会员
P2	个人 basic 会员
P3	个人 free 用户

管理员：队列统计

GET `/admin/v1/models/queue/stats`

管理员：指定模型队列详情

GET `/admin/v1/models/{model_id}/queue`

管理员：清空模型队列

POST `/admin/v1/models/{model_id}/queue/flush`

11.6 降级策略管理（管理员）

降级规则列表

GET `/admin/v1/models/fallback-rules`


```
{
  "code": 0,
  "data": {
    "items": [
      {
        "rule_id": "fb_001",
        "primary_model_id": "gpt-4o",
        "fallback_chain": [
          { "model_id": "gpt-4o-mini", "priority": 1 },
          { "model_id": "qwen-turbo",  "priority": 2 }
        ],
        "trigger_conditions": {
          "error_rate_threshold": 0.3,
          "timeout_ms": 30000,
          "consecutive_errors":
```

```
3
        },
        "notify_user": true,
        "status": "enabled"
      }
    ]
  }
}
```

创建降级规则

POST `/admin/v1/models/fallback-rules`

更新降级规则

PUT `/admin/v1/models/fallback-rules/{rule_id}`

删除降级规则

DELETE `/admin/v1/models/fallback-rules/{rule_id}`

降级触发日志

GET `/admin/v1/models/fallback-rules/logs`


```
{
  "code": 0,
  "data": {
    "items": [
      {
        "log_id": "fb_log_001",
        "rule_id": "fb_001",
        "primary_model_id": "gpt-4o",
        "used_model_id": "gpt-4o-mini",
        "trigger_reason": "consecutive_errors",
        "error_count": 3,
        "triggered_at": "2024-01-01T12:00:00Z"
      }
    ]
  }
}
```

10-token-api.md

Token 模块

Token 模块专注于 AI 模型调用中 input/output token 的用量追踪、平台定价管理和第三方市场价格对比，是计费模块的核心数据源。

10.1 用量仪表盘（核心看板）

总览仪表盘

GET `/api/v1/tokens/usage/dashboard`

查询参数	说明
`period`	`today` / `week` / `month` / `custom`
`start_date`	period=custom 时必填

查询参数	说明
`end_date`	period=custom 时必填


```
{
  "code": 0,
  "data": {
    "today": {
      "input_tokens": 128000,
      "output_tokens": 64000,
      "total_tokens": 192000,
      "total_cost": "5.38",
      "request_count": 350
    },
    "this_month": {
      "input_tokens": 3200000,
      "output_tokens": 1600000,
      "total_tokens": 4800000,
      "total_cost": "134.40",
      "request_count": 8750
    },
    "trend": [
      { "date": "2024-01-01",
"total_tokens": 180000, "total_
cost": "5.04", "request_count":
320 },
      { "date": "2024-01-02",
```

```
"total_tokens": 192000, "total_
cost": "5.38", "request_count":
350 }
    ]
  }
}
```


按 API Key 统计

GET `/api/v1/tokens/usage/by-apikey`


```
{
  "code": 0,
  "data": {
    "items": [
      {
        "key_id": "key_01H
X...",
        "key_name": "生产环境",
        "key_prefix": "sk-prod-
****",
        "request_count": 5200,
        "input_tokens": 210000
0,
        "output_tokens": 98000
0,
        "total_tokens": 308000
```

```
0,
        "total_cost": "86.24",
        "last_used_at": "2024-0
1-01T12:00:00Z"
      }
    ]
  }
}
```

Top 10 模型用量排行

GET `/api/v1/tokens/usage/top-models`

```
{
  "code": 0,
  "data": {
    "period": "this_month",
    "items": [
      {
        "rank": 1,
        "model_id": "gpt-4o",
        "display_name": "GPT-4
o",
        "provider_name": "OpenA
I",
        "request_count": 3200,
        "input_tokens": 150000
```

```
0,
        "output_tokens": 72000
0,
        "total_tokens": 222000
0,
        "total_cost": "62.16",
        "avg_latency_ms": 1240
      }
    ]
  }
}
```

Top 10 用户用量排行（管理员）

GET `/admin/v1/tokens/usage/top-users`

实时用量（最近 60 分钟）

GET `/api/v1/tokens/usage/realtime`

```
{
  "code": 0,
  "data": {
    "items": [
      { "minute": "2024-01-01T1
2:00:00Z", "total_tokens": 320
```

```
0, "request_count": 5 },
      { "minute": "2024-01-01T1
2:01:00Z", "total_tokens": 480
0, "request_count": 8 }
    ]
  }
}
```


用量热力图

GET
`/api/v1/tokens/usage/heatmap`


```
{
  "code": 0,
  "data": {
    "data": [
      { "weekday": "Monday", "h
our": 10, "total_tokens": 4500
0, "level": 3 },
      { "weekday": "Monday", "h
our": 11, "total_tokens": 8900
0, "level": 5 }
    ]
  }
}
```

`level` 范围 1-5，数值越高代表用量越高。

10.2 用量明细

总用量汇总

GET `/api/v1/tokens/usage`

用量明细列表

GET `/api/v1/tokens/usage/detail`

查询参数	说明
`model_id`	按模型筛选
`api_key_id`	按 API Key 筛选
`start_date`	开始日期
`end_date`	结束日期
`page`	页码

```
{
  "code": 0,
  "data": {
```

```
    "items": [
      {
        "request_id": "req_abc1
23",
        "model_id": "gpt-4o",
        "model_name": "GPT-4o",
        "api_key_id": "key_01H
X...",
        "input_tokens": 512,
        "output_tokens": 256,
        "total_tokens": 768,
        "input_cost": "0.0143
4",
        "output_cost": "0.0215
0",
        "total_cost": "0.0358
4",
        "latency_ms": 1180,
        "currency": "CNY",
        "created_at": "2024-01-
01T12:00:00Z"
      }
    ]
  }
}
```


每日用量趋势

GET `/api/v1/tokens/usage/daily`

月度用量汇总

GET
`/api/v1/tokens/usage/monthly`

单次请求 Token 详情

GET
`/api/v1/tokens/usage/{request_id}`

10.3 平台 Token 定价管理（管理员）

全量定价列表

GET `/admin/v1/tokens/pricing`


```
{
  "code": 0,
  "data": {
    "items": [
      {
        "pricing_id": "price_00
```

```
1",
        "model_id": "gpt-4o",
        "model_name": "GPT-4o",
        "currency": "CNY",
        "unit": "per_1k_token
s",
        "tiers": [
          { "membership_level":
"free",    "input_price": "0.03
5", "output_price": "0.105" },
          { "membership_level":
"basic",   "input_price": "0.02
8", "output_price": "0.084" },
          { "membership_level":
"premium", "input_price": "0.02
1", "output_price": "0.063" }
        ],
        "effective_at": "2024-0
1-01T00:00:00Z"
      }
    ]
  }
}
```


指定模型定价详情

GET

`/admin/v1/tokens/pricing/{model_`

`id}`

创建定价规则

POST `/admin/v1/tokens/pricing`

```
{
  "model_id": "claude-3-5-sonnet",
  "currency": "CNY",
  "unit": "per_1k_tokens",
  "tiers": [
    { "membership_level": "free",    "input_price": "0.045", "output_price": "0.135" },
    { "membership_level": "basic",   "input_price": "0.036", "output_price": "0.108" },
    { "membership_level": "premium", "input_price": "0.027", "output_price": "0.081" }
  ],
  "effective_at": "2024-02-01T00:00:00Z"
}
```

更新定价规则

PUT `/admin/v1/tokens/pricing/{pricing_id}`

删除定价规则

DELETE `/admin/v1/tokens/pricing/{pricing_id}`

定价历史变更记录

GET `/admin/v1/tokens/pricing/history/{model_id}`

10.4 第三方市场价格

市场价格列表

GET `/api/v1/tokens/market-prices`

查询参数	说明
`provider`	提供商 slug

查询参数	说明
`model_name`	模型名称


```
{
  "code": 0,
  "data": {
    "items": [
      {
        "id": "mkt_001",
        "provider": "openai",
        "provider_name": "OpenA
I",
        "model_name": "gpt-4o",
        "input_price": "0.037
5",
        "output_price": "0.112
5",
        "currency": "CNY",
        "source_url": "https://
openai.com/pricing",
        "updated_at": "2024-01-
01T00:00:00Z"
      }
    ]
  }
}
```

添加第三方价格（管理员）

POST `/admin/v1/tokens/market-prices`

更新第三方价格（管理员）

PUT `/admin/v1/tokens/market-prices/{id}`

删除第三方价格（管理员）

DELETE `/admin/v1/tokens/market-prices/{id}`

10.5 价格对比分析

全模型价格对比

GET `/api/v1/tokens/price-comparison`

```
{
  "code": 0,
  "data": {
    "items": [
```

```
      {
        "model_id": "gpt-4o",
        "display_name": "GPT-4
o",
        "platform_price": {
          "input": "0.028",
          "output": "0.084",
          "membership_level":
"basic"
        },
        "market_prices": [
          {
            "provider": "opena
i",
            "input": "0.0375",
            "output": "0.1125",
            "savings_input": "2
5.3%",
            "savings_output":
"25.3%"
          }
        ]
      }
    ]
  }
}
```


指定模型价格对比

GET `/api/v1/tokens/price-comparison/{model_id}`

10.6 费用预估

POST `/api/v1/tokens/estimate`

```
{
  "model_id": "gpt-4o",
  "input_tokens": 1000,
  "output_tokens": 500
}
```

响应：

```
{
  "code": 0,
  "data": {
    "model_id": "gpt-4o",
    "input_tokens": 1000,
    "output_tokens": 500,
    "input_cost": "0.028",
    "output_cost": "0.042",
    "total_cost": "0.070",
    "currency": "CNY",
    "applied_pricing": {
      "membership_level": "basi
```

```
c",
      "input_price": "0.028",
      "output_price": "0.084"
    }
  }
}
```

09-gateway-api.md

API 网关

API 网关负责统一路由、全局限流、访问日志、Webhook 事件推送和审计日志，所有接口均为管理员专用。

9.1 路由管理

路由规则列表

GET `/admin/v1/gateway/routes`


```
{
  "code": 0,
  "data": {
    "items": [
      {
        "route_id": "route_00
1",
        "name": "AI 网关路由",
        "path_prefix": "/api/v
1/chat/",
        "upstream": "http://ai-
```

```
gateway:8001",
        "methods": ["POST"],
        "status": "active",
        "created_at": "2024-01-
01T00:00:00Z"
      }
    ]
  }
}
```

添加路由规则

POST `/admin/v1/gateway/routes`

```
{
  "name": "业务后端路由",
  "path_prefix": "/api/v1/",
  "upstream": "http://business:
8000",
  "methods": ["GET", "POST", "P
UT", "DELETE"],
  "strip_prefix": false,
  "timeout_ms": 30000
}
```

更新路由规则

PUT `/admin/v1/gateway/routes/{route_id}`

删除路由规则

DELETE `/admin/v1/gateway/routes/{route_id}`

9.2 全局限流

限流规则列表

GET `/admin/v1/gateway/rate-limits`


```
{
  "code": 0,
  "data": {
    "items": [
      {
        "limit_id": "rl_global_001",
        "name": "全局 IP 限流",
        "dimension": "ip",
        "requests_per_minute":
```

```
300,
        "requests_per_day": 500
00,
        "action": "reject",
        "status": "active"
      }
    ]
  }
}
```

创建限流规则

POST `/admin/v1/gateway/rate-limits`

```
{
  "name": "API Key 每分钟限流",
  "dimension": "api_key",
  "requests_per_minute": 60,
  "requests_per_day": 10000,
  "action": "reject"
}
```

`dimension` 枚举：`ip` / `api_key` / `user_id`
`action` 枚举：`reject` / `queue`

更新限流规则

PUT `/admin/v1/gateway/rate-limits/{limit_id}`

删除限流规则

DELETE `/admin/v1/gateway/rate-limits/{limit_id}`

9.3 访问日志

访问日志列表

GET `/admin/v1/gateway/logs`

查询参数	说明
`api_key_id`	按 API Key 筛选
`path`	按接口路径筛选
`status_code`	按 HTTP 状态码筛选
`start_time`	开始时间
`end_time`	结束时间

查询参数	说明
`page`	页码
`page_size`	每页条数


```
{
  "code": 0,
  "data": {
    "items": [
      {
        "log_id": "log_01H
X...",
        "method": "POST",
        "path": "/api/v1/chat/c
ompletions",
        "status_code": 200,
        "latency_ms": 1240,
        "api_key_id": "key_01H
X...",
        "user_id": "u_01HX...",
        "ip": "1.2.3.4",
        "request_size": 1024,
        "response_size": 4096,
        "created_at": "2024-01-
01T12:00:00Z"
      }
    ],
    "total": 125000
```

```
  }
}
```

指定 API Key 流量统计

GET `/admin/v1/gateway/stats/apikey/{key_id}`

```
{
  "code": 0,
  "data": {
    "key_id": "key_01HX...",
    "period": "today",
    "request_count": 1250,
    "success_count": 1230,
    "error_count": 20,
    "success_rate": "98.4%",
    "avg_latency_ms": 980,
    "p99_latency_ms": 3200
  }
}
```

网关流量统计

GET `/admin/v1/gateway/stats`

```
{
  "code": 0,
  "data": {
    "period": "today",
    "total_requests": 128500,
    "success_rate": "99.1%",
    "avg_qps": 1.49,
    "peak_qps": 45.2,
    "avg_latency_ms": 890,
    "p99_latency_ms": 2800,
    "rate_limited_count": 234
  }
}
```

9.4 Webhook 管理

Webhook 端点列表

GET `/admin/v1/gateway/webhooks`


```
{
  "code": 0,
  "data": {
    "items": [
      {
        "webhook_id": "wh_01H
```

```
X...",
        "name": "充值成功通知",
        "url": "https://partne
r.example.com/webhooks/recharg
e",
        "event_types": ["rechar
ge.success", "balance.low"],
        "status": "active",
        "secret": "whsec_****",
        "created_at": "2024-01-
01T00:00:00Z"
      }
    ]
  }
}
```


创建 Webhook

POST `/admin/v1/gateway/webhooks`


```
{
  "name": "充值成功通知",
  "url": "https://partner.examp
le.com/webhooks/recharge",
  "event_types": ["recharge.suc
cess", "balance.low"],
```

```
  "secret": "my_webhook_secret"
}
```

可订阅的事件类型（event_types）：

事件类型	触发时机
`recharge.success`	充值到账
`balance.low`	余额低于
`membership.expired`	会员到期
`apikey.expired`	API Key 到
`usage.quota_exceeded`	用量超配
`model.degraded`	模型发生
`order.paid`	订单支付
`order.refunded`	订单退款

更新 Webhook

PUT `/admin/v1/gateway/webhooks/{webhook_id}`

删除 Webhook

DELETE `/admin/v1/gateway/webhooks/{webhook_id}`

发送测试事件

POST `/admin/v1/gateway/webhooks/{webhook_id}/test`

```
{ "event_type": "recharge.success" }
```

Webhook 投递日志

GET `/admin/v1/gateway/webhooks/{webhook_id}/logs`

```
{
  "code": 0,
  "data": {
    "items": [
      {
        "log_id": "whl_001",
        "event_type": "recharge.success",
        "status": "success",
```

```
        "response_code": 200,
        "latency_ms": 245,
        "retry_count": 0,
        "sent_at": "2024-01-01T
12:05:00Z"
      }
    ]
  }
}
```

9.5 审计日志

GET `/admin/v1/gateway/audit-logs`

查询参数	说明
`user_id`	按用户筛选
`action`	按操作类型筛选
`start_time`	开始时间
`end_time`	结束时间

```
{
  "code": 0,
  "data": {
```

```
    "items": [
      {
        "log_id": "audit_01H
X...",
        "user_id": "u_01HX...",
        "user_email": "user@exa
mple.com",
        "action": "apikey.delet
e",
        "resource_type": "api_k
ey",
        "resource_id": "key_01H
X...",
        "ip": "1.2.3.4",
        "user_agent": "Mozilla/
5.0...",
        "result": "success",
        "detail": { "key_name":
"生产环境" },
        "created_at": "2024-01-
01T12:00:00Z"
      }
    ]
  }
}
```

常见操作类型（action）：

值	说明
`user.login`	用户登录
`user.logout`	用户登出
`user.password_change`	修改密码
`apikey.create`	创建 AP
`apikey.delete`	删除 AP
`enterprise.sso_config`	修改 SS 配置
`admin.user_freeze`	冻结用户
`admin.balance_adjust`	手动调整

08-member-api.md

会员模块

8.1 套餐管理

会员套餐列表

GET `/api/v1/membership/plans`

查询参数	说明
`type`	`personal` / `enterprise`
`status`	`active` / `disabled`


```
{
  "code": 0,
  "data": {
    "items": [
      {
        "plan_id": "plan_basic",
        "name": "基础版",
```

```
        "type": "personal",
        "price_monthly": "49.0
0",
        "price_yearly": "490.0
0",
        "currency": "CNY",
        "status": "active",
        "benefits": [
          { "key": "tpm",
"value": 100000,
"desc": "每分钟 10万 Token" },
          { "key": "tpd",
"value": 2000000,
"desc": "每日 200万 Token" },
          { "key": "models",
"value": ["gpt-4o", "qwen-turb
o"],    "desc": "可用模型" },
          { "key": "token_disco
unt", "value": "20%",
"desc": "Token 费率 8折" },
          { "key": "priority",
"value": "P1",
"desc": "队列优先级 P1" },
          { "key": "support",
"value": "email",
"desc": "邮件工单支持" }
        ]
      },
      {
```

```
        "plan_id": "plan_premium",
        "name": "高级版",
        "type": "personal",
        "price_monthly": "199.00",
        "price_yearly": "1990.00",
        "currency": "CNY",
        "benefits": [
          { "key": "tpm", "value": 500000,    "desc": "每分钟 50万 Token" },
          { "key": "tpd", "value": 10000000,  "desc": "每日 1000万 Token" },
          { "key": "token_discount", "value": "40%",     "desc": "Token 费率 6折" },
          { "key": "priority", "value": "P1",      "desc": "队列优先级 P1" },
          { "key": "support", "value": "priority","desc": "优先技术支持" }
        ]
      }
    ]
```

```
  }
}
```

套餐详情

GET `/api/v1/membership/plans/{plan_id}`

创建套餐（管理员）

POST `/admin/v1/membership/plans`

```
{
  "name": "企业标准版",
  "type": "enterprise",
  "price_monthly": "999.00",
  "price_yearly": "9990.00",
  "currency": "CNY",
  "benefits": [
    { "key": "tpm", "value": 2000000, "desc": "每分钟 200万 Token" },
    { "key": "member_seats", "value": 50, "desc": "50个成员席位" },
    { "key": "priority", "valu
```

```
e": "P0", "desc": "队列最高优先级" }
  ]
}
```


更新套餐（管理员）

PUT `/admin/v1/membership/plans/{plan_id}`

8.2 我的会员状态

GET `/api/v1/membership/me`


```
{
  "code": 0,
  "data": {
    "plan_id": "plan_basic",
    "plan_name": "基础版",
    "plan_type": "personal",
    "status": "active",
    "period": "yearly",
    "started_at": "2024-01-01T00:00:00Z",
    "expired_at": "2024-12-31T2
```

```
3:59:59Z",
    "auto_renew": true,
    "usage_this_month": {
      "tokens_used": 1200000,
      "tokens_limit": 2000000,
      "usage_percent": 60
    },
    "effective_benefits": {
      "tpm": 100000,
      "token_discount": "20%",
      "priority": "P1"
    }
  }
}
```

`status` 枚举：`active` / `expired` / `cancelled` / `none`（未订阅）

8.3 订阅会员

POST `/api/v1/membership/subscribe`

```
{
  "plan_id": "plan_basic",
  "period": "yearly",
```

```
  "auto_renew": true,
  "voucher_id": "v_01HX..."
}
```

响应： 创建订单，返回支付参数（同订单模块）

```
{
  "code": 0,
  "data": {
    "order_id": "ord_20240101010",
    "plan_name": "基础版（年付）",
    "original_amount": "490.00",
    "discount_amount": "20.00",
    "final_amount": "470.00",
    "currency": "CNY"
  }
}
```

8.4 续费

POST `/api/v1/membership/renew`

```
{
  "period": "yearly",
  "voucher_id": null
}
```

8.5 取消自动续费

POST `/api/v1/membership/cancel`

取消后当前会员期结束后不再自动扣款，当前权益保留至到期日。

```
{ "reason": "暂时不需要了" }
```

响应：

```
{
  "code": 0,
  "data": {
    "status": "cancelled",
    "auto_renew": false,
    "valid_until": "2024-12-31T23:59:59Z",
    "message": "您的会员权益将保留至 2024-12-31"
```

```
  }
}
```

8.6 会员变更历史

GET `/api/v1/membership/history`

```
{
  "code": 0,
  "data": {
    "items": [
      {
        "history_id": "mh_001",
        "action": "subscribe",
        "plan_name": "基础版",
        "period": "yearly",
        "amount": "490.00",
        "started_at": "2024-01-01T00:00:00Z",
        "expired_at": "2024-12-31T23:59:59Z",
        "order_id": "ord_20240101010",
        "created_at": "2024-01-01T00:00:00Z"
      }
    ]
```

```
  }
}
```

`action` 枚举：`subscribe` / `renew` / `upgrade` / `downgrade` / `cancel` / `expired`

07-accounting-api.md

账务模块

账务模块面向财务人员，提供收支报表、总账流水、对账管理和增值税发票审核功能。

7.1 收支报表

月度收支报表

GET
`/api/v1/accounting/statements`

查询参数	说明
`year`	年份，如 `2024`
`month`	月份，如 `1`（可选，不传则返回全年）

```
{
  "code": 0,
  "data": {
```

```
    "items": [
      {
        "period": "2024-01",
        "income": {
          "recharge": "5000.0
0",
          "gift": "200.00",
          "total": "5200.00"
        },
        "expense": {
          "consume": "4200.00",
          "refund_out": "50.0
0",
          "total": "4250.00"
        },
        "net": "950.00",
        "currency": "CNY"
      }
    ]
  }
}
```


导出收支报表

GET `/api/v1/accounting/statements/export`

查询参数	说明
`year`	年份
`format`	`csv` / `xlsx`

7.2 总账流水

GET `/api/v1/accounting/ledger`

查询参数	说明
`start_date`	开始日期
`end_date`	结束日期
`type`	资金变动类型
`page`	页码
`page_size`	每页条数


```
{
  "code": 0,
  "data": {
    "items": [
      {
        "ledger_id": "ldg_001",
        "type": "recharge",
```

```
        "amount": "+100.00",
        "user_id": "u_01HX...",
        "description": "用户充值 - 支付宝",
        "created_at": "2024-01-01T12:05:00Z"
      }
    ],
    "total": 15234
  }
}
```

7.3 对账管理（管理员）

对账记录列表

GET `/admin/v1/accounting/reconciliation`

```
{
  "code": 0,
  "data": {
    "items": [
      {
        "recon_id": "recon_2024
```

```
0101",
        "period": "2024-01-01",
        "channel": "alipay",
        "status": "matched",
        "platform_count": 125,
        "channel_count": 125,
        "diff_count": 0,
        "initiated_at": "2024-0
1-02T02:00:00Z",
        "completed_at": "2024-0
1-02T02:05:00Z"
      }
    ]
  }
}
```

发起对账

POST `/admin/v1/accounting/reconciliat
ion`

```
{
  "period": "2024-01-01",
  "channel": "alipay"
}
```

`channel` 枚举：`alipay` / `wechat` / `baofoo`

响应：

```
{
  "code": 0,
  "data": {
    "recon_id": "recon_20240101",
    "status": "processing",
    "estimated_seconds": 60
  }
}
```

对账详情

GET `/admin/v1/accounting/reconciliation/{recon_id}`

```
{
  "code": 0,
  "data": {
    "recon_id": "recon_20240101",
    "period": "2024-01-01",
    "channel": "alipay",
    "status": "diff_found",
```

```
    "platform_count": 125,
    "channel_count": 124,
    "diff_count": 1,
    "diffs": [
      {
        "order_id": "rch_202401
01099",
        "platform_status": "pai
d",
        "channel_status": "not_
found",
        "amount": "200.00",
        "diff_type": "platform_
only"
      }
    ]
  }
}
```

确认对账结果

PUT `/admin/v1/accounting/reconciliat
ion/{recon_id}/confirm`

```
{
  "action": "accept",
  "note": "已核实，支付平台数据
```

```
延迟"
}
```

7.4 增值税发票管理（管理员）

开票申请列表

GET `/admin/v1/accounting/vat-invoices`

查询参数	说明
`status`	`pending` / `processing` / `completed` / `rejected`
`start_date`	开始日期

```
{
  "code": 0,
  "data": {
    "items": [
      {
        "invoice_id": "inv_req_
```

```
01HX...",
        "user_id": "u_01HX...",
        "enterprise_name": "XX
科技有限公司",
        "invoice_type": "vat_sp
ecial",
        "total_amount": "500.0
0",
        "status": "pending",
        "submitted_at": "2024-0
1-05T10:00:00Z"
      }
    ]
  }
}
```

处理开票申请（通过）

PUT `/admin/v1/accounting/vat-invoices/{invoice_id}/process`

```
{
  "invoice_number": "123456789
0",
  "invoice_file_url": "https://
oss.example.com/invoices/xxx.pd
```

```
f"
}
```

拒绝开票申请

PUT `/admin/v1/accounting/vat-invoices/{invoice_id}/reject`

```
{
  "reason": "税号填写有误，请重新提交"
}
```

06-billing-api.md

计费模块

计费模块管理模型调用的费用规则、用量汇总及月度账单，是 Token 模块用量数据的财务映射。

6.1 计费规则

查询计费规则列表

GET `/api/v1/billing/rules`

查询参数	说明
`model_id`	按模型筛选
`membership_level`	按会员等级筛

```
{
  "code": 0,
  "data": {
    "items": [
```

```
      {
        "rule_id": "br_001",
        "model_id": "gpt-4o",
        "model_name": "GPT-4o",
        "membership_level": "ba
sic",
        "input_price": "0.028",
        "output_price": "0.08
4",
        "unit": "per_1k_token
s",
        "currency": "CNY",
        "effective_at": "2024-0
1-01T00:00:00Z"
      }
    ]
  }
}
```

查询计费规则详情

GET `/api/v1/billing/rules/{rule_id}`

创建计费规则（管理员）

POST `/admin/v1/billing/rules`

```
{
  "model_id": "claude-3-5-sonnet",
  "membership_level": "free",
  "input_price": "0.045",
  "output_price": "0.135",
  "unit": "per_1k_tokens",
  "currency": "CNY",
  "effective_at": "2024-02-01T00:00:00Z"
}
```


更新计费规则（管理员）

PUT `/admin/v1/billing/rules/{rule_id}`

6.2 费用预估

POST `/api/v1/billing/estimate`


```
{
  "model_id": "gpt-4o",
  "input_tokens": 1000,
```

```
  "output_tokens": 500
}
```

响应：

```
{
  "code": 0,
  "data": {
    "model_id": "gpt-4o",
    "model_name": "GPT-4o",
    "input_tokens": 1000,
    "output_tokens": 500,
    "input_cost": "0.028",
    "output_cost": "0.042",
    "total_cost": "0.070",
    "currency": "CNY",
    "applied_rule": {
      "membership_level": "basi
c",
      "input_price": "0.028",
      "output_price": "0.084"
    }
  }
}
```

6.3 用量统计

用量汇总

GET `/api/v1/billing/usage`

查询参数	说明
`period`	`today` / `week` / `month` / `custom`
`start_date`	period=custom 必填
`end_date`	period=custom 必填


```
{
  "code": 0,
  "data": {
    "period": "month",
    "input_tokens": 3200000,
    "output_tokens": 1600000,
    "total_tokens": 4800000,
    "request_count": 8750,
    "total_cost": "134.40",
    "currency": "CNY"
  }
}
```

每日用量明细

GET `/api/v1/billing/usage/daily`

```
{
  "code": 0,
  "data": {
    "items": [
      {
        "date": "2024-01-01",
        "input_tokens": 128000,
        "output_tokens": 64000,
        "total_tokens": 192000,
        "request_count": 350,
        "total_cost": "5.38"
      }
    ]
  }
}
```

按模型分组费用

GET `/api/v1/billing/usage/by-model`

```
{
  "code": 0,
  "data": {
```

```
    "items": [
      {
        "model_id": "gpt-4o",
        "model_name": "GPT-4o",
        "input_tokens": 2000000,
        "output_tokens": 960000,
        "total_cost": "89.60",
        "request_count": 5200
      }
    ]
  }
}
```

按 API Key 分组费用

GET `/api/v1/billing/usage/by-apikey`

```
{
  "code": 0,
  "data": {
    "items": [
      {
        "key_id": "key_01HX...",
        "key_name": "生产环境",
```

```
        "key_prefix": "sk-prod-
****",
        "total_tokens": 308000
0,
        "total_cost": "86.24",
        "request_count": 5200
      }
    ]
  }
}
```

6.4 月度账单

账单列表

GET `/api/v1/billing/invoices`

```
{
  "code": 0,
  "data": {
    "items": [
      {
        "invoice_id": "inv_2024
01",
        "period": "2024-01",
        "total_tokens": 480000
```

```
0,
        "total_cost": "134.40",
        "currency": "CNY",
        "status": "settled",
        "settled_at": "2024-02-
01T00:00:00Z"
      }
    ]
  }
}
```

账单详情

GET `/api/v1/billing/invoices/{invoice_id}`

```
{
  "code": 0,
  "data": {
    "invoice_id": "inv_202401",
    "period": "2024-01",
    "total_tokens": 4800000,
    "total_cost": "134.40",
    "currency": "CNY",
    "status": "settled",
    "breakdown": [
      {
```

```
        "model_id": "gpt-4o",
        "model_name": "GPT-4o",
        "input_tokens": 2000000,
        "output_tokens": 960000,
        "total_cost": "89.60"
      },
      {
        "model_id": "qwen-turbo",
        "model_name": "通义千问 Turbo",
        "input_tokens": 1200000,
        "output_tokens": 640000,
        "total_cost": "44.80"
      }
    ],
    "settled_at": "2024-02-01T00:00:00Z"
  }
}
```


下载账单 PDF

GET `/api/v1/billing/invoices/{invoice_id}/download`

```
{
  "code": 0,
  "data": {
    "download_url": "https://oss.example.com/invoices/inv_202401.pdf",
    "expires_at": "2024-02-01T12:10:00Z"
  }
}
```

05-order-api.md

订单模块

订单是发起支付的载体，支持充值订单和会员套餐购买订单两种类型。

订单状态流转

```
pending（待支付）
  ├──── → paid（已支付）→ refunding（退款中）→ refunded（已退款）
  └──── → cancelled（已取消）/ expired（已过期）
```

5.1 订单列表

GET `/api/v1/orders`

查询参数	类型	说明
`type`	string	`recharge` `membershi`
`status`	string	`pending` / `paid` / `cancelled` / `refunding` / `refunded` / `expired`
`start_date`	string	开始日期
`end_date`	string	结束日期
`page`	int	页码
`page_size`	int	每页条数

响应：

```
{
  "code": 0,
  "data": {
    "items": [
      {
        "order_id": "ord_20240101001",
```

```
        "type": "recharge",
        "status": "paid",
        "amount": "100.00",
        "currency": "CNY",
        "channel": "alipay",
        "description": "钱包充值 100 元",
        "paid_at": "2024-01-01T12:05:00Z",
        "created_at": "2024-01-01T12:00:00Z"
      }
    ],
    "total": 12,
    "page": 1,
    "page_size": 20
  }
}
```

5.2 创建订单

POST `/api/v1/orders`

```
{
  "type": "membership",
  "plan_id": "plan_basic",
  "period": "yearly",
```

```
  "voucher_id": "v_01HX..."
}
```

字段	类型	必填
`type`	string	是
`amount`	string	recharge 必填
`plan_id`	string	membership 必填
`period`	string	membership 必填
`voucher_id`	string	否

响应：

```
{
  "code": 0,
  "data": {
    "order_id": "ord_20240101002",
    "type": "membership",
    "status": "pending",
    "amount": "490.00",
```

```
    "discount_amount": "20.00",
    "final_amount": "470.00",
    "currency": "CNY",
    "expire_at": "2024-01-01T12:30:00Z"
  }
}
```

5.3 订单详情

GET `/api/v1/orders/{order_id}`

```
{
  "code": 0,
  "data": {
    "order_id": "ord_20240101001",
    "type": "recharge",
    "status": "paid",
    "amount": "100.00",
    "discount_amount": "0.00",
    "final_amount": "100.00",
    "currency": "CNY",
    "channel": "alipay",
    "channel_trade_no": "20240101220012001XXXX",
    "description": "钱包充值 10
```

```
0 元",
    "voucher_id": null,
    "paid_at": "2024-01-01T12:05:00Z",
    "created_at": "2024-01-01T12:00:00Z",
    "expire_at": "2024-01-01T12:30:00Z"
  }
}
```

5.4 发起支付

POST `/api/v1/orders/{order_id}/pay`

```
{
  "channel": "wechat",
  "client_type": "web",
  "return_url": "https://console.example.com/membership"
}
```

响应（微信扫码）：

```
{
  "code": 0,
  "data": {
    "order_id": "ord_20240101002",
    "channel": "wechat",
    "pay_params": {
      "type": "qrcode",
      "code_url": "weixin://wxpay/bizpayurl?pr=xxx"
    },
    "expire_at": "2024-01-01T12:30:00Z"
  }
}
```

5.5 查询支付状态

GET `/api/v1/orders/{order_id}/payment-status`

前端可轮询此接口（建议间隔 2 秒），直到 `status` 为 `paid` 或超时。

```
{
  "code": 0,
  "data": {
    "order_id": "ord_20240101002",
    "status": "paid",
    "paid_at": "2024-01-01T12:08:00Z"
  }
}
```

5.6 取消订单

POST `/api/v1/orders/{order_id}/cancel`

仅 `pending` 状态的订单可取消。

```
{ "reason": "不想买了" }
```

5.7 申请退款

POST `/api/v1/orders/{order_id}/refund`

仅 `paid` 状态且满足退款条件的订单可申请。

```
{
  "reason": "误操作重复购买",
  "amount": "100.00"
}
```

响应：

```
{
  "code": 0,
  "data": {
    "refund_id": "ref_01HX...",
    "order_id": "ord_20240101001",
    "refund_amount": "100.00",
    "status": "processing",
    "estimated_days": 3,
    "created_at": "2024-01-01T15:00:00Z"
  }
}
```

5.8 导出订单

GET `/api/v1/orders/export`

查询参数	说明
`format`	`csv` / `xlsx`
`start_date`	开始日期
`end_date`	结束日期
`type`	订单类型筛选
`status`	订单状态筛选

响应： 返回文件下载链接（有效期 10 分钟）

04-transaction-api.md

交易模块

交易模块记录账户所有资金变动明细，是钱包余额每一次变化的完整历史凭证。

交易类型说明

type	说明	金额
`recharge`	充值入账	正
`consume`	模型调用消费	负
`refund`	退款到账	正
`gift`	赠送（代金券/活动）	正
`freeze`	余额冻结	负
`unfreeze`	余额解冻	正
`adjust`	管理员手动调整	正或

4.1 交易记录列表

GET `/api/v1/transactions`

查询参数

参数	类型	必填	说明
`type`	string	否	交易 支持
`start_date`	string	否	开始
`end_date`	string	否	结束
`page`	int	否	页码
`page_size`	int	否	每页
`order`	string	否	排序 `des`

响应

```
{
  "code": 0,
  "data": {
    "items": [
```

```
      {
        "tx_id": "tx_20240101001",
        "type": "consume",
        "amount": "-0.038",
        "balance_after": "148.46",
        "gift_balance_after": "20.00",
        "currency": "CNY",
        "description": "调用 gpt-4o, 768 tokens",
        "related_id": "req_abc123",
        "related_type": "model_request",
        "created_at": "2024-01-01T12:00:00Z"
      },
      {
        "tx_id": "tx_20240101000",
        "type": "recharge",
        "amount": "+100.00",
        "balance_after": "148.50",
        "gift_balance_after": "20.00",
        "currency": "CNY",
```

```
        "description": "支付宝
充值",
        "related_id": "rch_2024
0101001",
        "related_type": "rechar
ge_order",
        "created_at": "2024-01-
01T11:55:00Z"
      }
    ],
    "total": 256,
    "page": 1,
    "page_size": 20,
    "total_pages": 13
  }
}
```

响应字段说明

字段	说明
`tx_id`	交易流水号
`type`	交易类型
`amount`	变动金额，负数为支出
`balance_after`	本次交易后

字段	说明
`gift_balance_after`	本次交易后
`related_id`	关联业务 ID 等）
`related_type`	关联业务类 / `recharge` `refund_or` `manual_ad`

HTTP 状态码： `200` 成功， `401` 未认证

4.2 交易详情

GET `/api/v1/transactions/{tx_id}`

```
{
  "code": 0,
  "data": {
    "tx_id": "tx_20240101001",
    "type": "consume",
    "amount": "-0.038",
    "balance_before": "148.50",
    "balance_after": "148.46",
```

```
    "gift_balance_before": "20.
00",
    "gift_balance_after": "20.0
0",
    "currency": "CNY",
    "description": "调用 gpt-4
o, 768 tokens",
    "related_id": "req_abc123",
    "related_type": "model_requ
est",
    "model_id": "gpt-4o",
    "input_tokens": 512,
    "output_tokens": 256,
    "total_tokens": 768,
    "api_key_id": "key_01H
X...",
    "api_key_name": "生产环境",
    "created_at": "2024-01-01T1
2:00:00Z"
  }
}
```


HTTP 状态码： `200` 成功， `404` 交易记录不存在

4.3 交易统计

GET `/api/v1/transactions/stats`

查询参数

参数	类型	必填
`period`	string	否
`start_date`	string	`custom` 时必填
`end_date`	string	`custom` 时必填
`group_by`	string	否

响应

```
{
  "code": 0,
  "data": {
    "period": "month",
    "summary": {
```

```
      "total_recharge": "500.0
0",
      "total_consume": "423.6
0",
      "total_gift": "20.00",
      "total_refund": "0.00",
      "net_change": "+96.40",
      "transaction_count": 312
    },
    "trend": [
      { "date": "2024-01-01",
"consume": "15.20", "recharge":
"100.00", "gift": "20.00" },
      { "date": "2024-01-02",
"consume": "8.40",  "recharge":
"0.00",   "gift": "0.00"  },
      { "date": "2024-01-03",
"consume": "12.80", "recharge":
"0.00",   "gift": "0.00"  }
    ]
  }
}
```

4.4 导出交易记录

GET `/api/v1/transactions/export`

查询参数

参数	类型	必填	说明
`format`	string	否	文件 / `x` `csv`
`start_date`	string	否	开始
`end_date`	string	否	结束
`type`	string	否	交易

导出为异步任务，响应返回下载链接（有效期 10 分钟）。

响应

```
{
  "code": 0,
  "data": {
    "download_url": "https://oss.example.com/exports/tx_export_u001_20240101.csv",
    "expires_at": "2024-01-01T12:10:00Z",
    "record_count": 256,
    "file_size": "48KB"
```

```
  }
}
```

HTTP 状态码： `200` 成功， `400` 参数错误（如时间范围超过 1 年）

4.5 管理员查询（管理端）

GET `/admin/v1/transactions`

在用户侧查询参数基础上，额外支持以下筛选：

参数	类型	说明
`user_id`	string	按指定用户
`api_key_id`	string	按指定 API K 筛选
`min_amount`	string	最小金额（绝对值）
`max_amount`	string	最大金额（绝对值）

GET `/admin/v1/transactions/export`

管理员导出支持全平台数据，可追加 `user_id` 参数导出指定用户的交易记录。

03-wallet-api.md

钱包模块

3.1 余额查询

查询账户余额

GET `/api/v1/wallet/balance`

```
{
  "code": 0,
  "data": {
    "balance": "128.50",
    "gift_balance": "20.00",
    "total_balance": "148.50",
    "frozen_balance": "0.00",
    "currency": "CNY"
  }
}
```

字段	说明
`balance`	实账余额（充值获得，可提现）
`gift_balance`	赠送余额（代金活动赠送，不可提现）
`total_balance`	可用总余额
`frozen_balance`	冻结余额（处理中的退款

余额变动历史

GET `/api/v1/wallet/balance/history`

查询参数	类型	说明
`type`	string	`recharge` / `consume` / `gift` / `refund`
`start_date`	string	开始日期 `YYYY-MM-`

查询参数	类型	说明
		`DD`
`end_date`	string	结束日期
`page`	int	页码
`page_size`	int	每页条数

3.2 充值

发起充值

POST `/api/v1/wallet/recharge`

```
{
  "amount": "100.00",
  "channel": "alipay",
  "return_url": "https://console.example.com/wallet",
  "client_type": "web"
}
```

字段	类型	必填	说明
`amount`	string	是	充值金额，最低 1.00

`channel`	string	是	`alipay`
			`wechat`
			`balance`

`return_url`	string	Web 必填	支付完成后跳转地址
`client_type`	string	是	`web` `mini_program`
`voucher_id`	string	否	使用代金券

响应（支付宝 Web）：

```
{
  "code": 0,
  "data": {
    "order_id": "rch_20240101001",
    "channel": "alipay",
    "amount": "100.00",
    "pay_params": {
      "type": "form",
      "form_html": "<form action='https://openapi.alipay.co
```

```
m/...'>"
    },
    "expire_at": "2024-01-01T1
2:30:00Z"
  }
}
```

响应（微信 Web 扫码）：

```
{
  "code": 0,
  "data": {
    "order_id": "rch_2024010100
2",
    "channel": "wechat",
    "amount": "100.00",
    "pay_params": {
      "type": "qrcode",
      "code_url": "weixin://wxp
ay/bizpayurl?pr=xxx"
    },
    "expire_at": "2024-01-01T1
2:30:00Z"
  }
}
```

查询充值订单状态

GET `/api/v1/wallet/recharge/{order_id}`

```
{
  "code": 0,
  "data": {
    "order_id": "rch_20240101001",
    "status": "paid",
    "amount": "100.00",
    "channel": "alipay",
    "paid_at": "2024-01-01T12:05:00Z"
  }
}
```

`status` 枚举：`pending` / `paid` / `cancelled` / `expired`

支付平台异步回调

POST `/api/v1/wallet/recharge/notify/{channel}`

由支付平台主动回调，无需前端调用。系统验签后更新订单状态并充值。

`channel` 枚举：`alipay` / `wechat` / `baofoo`

3.3 代金券

查询代金券列表

GET `/api/v1/wallet/vouchers`

查询参数	说明
`status`	`available` / `used` / `expired`
`page`	页码


```
{
  "code": 0,
  "data": {
    "items": [
      {
        "voucher_id": "v_01HX...",
        "name": "新用户注册礼包",
        "voucher_type": "fixed_amount",
```

```
        "amount": "20.00",
        "min_recharge": "0.00",
        "status": "available",
        "applicable_models":
[],
        "valid_until": "2024-12
-31T23:59:59Z",
        "redeemed_at": null,
        "used_at": null
      }
    ],
    "total": 3
  }
}
```

兑换代金券

POST `/api/v1/wallet/vouchers/redeem`

```
{ "code": "VOUCHER2024001" }
```

响应：

```
{
  "code": 0,
  "data": {
    "voucher_id": "v_01HX...",
```

```
    "name": "新用户注册礼包",
    "amount": "20.00",
    "valid_until": "2024-12-31T
23:59:59Z"
  }
}
```

错误码： `3101` 券码无效， `3102` 已使用， `3103` 已过期

查询代金券详情

GET `/api/v1/wallet/vouchers/{voucher_id}`

3.4 发票申请

申请开票

POST `/api/v1/wallet/invoices`

```
{
  "order_ids": ["rch_2024010100
1", "rch_20240102001"],
  "invoice_type": "vat_specia
```

```
l",
  "title": "XX 科技有限公司",
  "tax_number": "91110000XXXXXX
XX",
  "bank_name": "招商银行北京分
行",
  "bank_account": "0000 0000 00
00 0000",
  "company_address": "北京市朝
阳区XX路1号",
  "company_phone": "010-8888888
8",
  "email": "finance@company.co
m"
}
```

字段	说明
`invoice_type`	`vat_special` 增值税专用发票 / `vat_general` 增值税普通发票
`order_ids`	要开票的充值订单 ID 列表（需为已支付状态

响应：

```
{
  "code": 0,
  "data": {
    "invoice_id": "inv_req_01HX...",
    "status": "pending",
    "total_amount": "200.00",
    "submitted_at": "2024-01-01T12:00:00Z"
  }
}
```


查询发票申请列表

GET `/api/v1/wallet/invoices`

查询参数	说明
`status`	`pending` / `processing` / `completed` / `rejected`

查询发票详情

GET `/api/v1/wallet/invoices/{invoice_id}`

```
{
  "code": 0,
  "data": {
    "invoice_id": "inv_req_01HX...",
    "status": "completed",
    "invoice_number": "XXXXXXXXXXX",
    "download_url": "https://oss.example.com/invoices/xxx.pdf",
    "completed_at": "2024-01-03T10:00:00Z"
  }
}
```

02-auth-api.md

授权模块

授权模块涵盖账号登录、Token 生命周期、第三方平台 OAuth 登录/绑定、企业 SSO，以及 RBAC 角色权限控制。

2.1 账号登录与 Token 管理

获取 RSA 公钥

所有涉及密码传输的接口（登录、注册、修改密码、重置密码）均不得明文传输密码。客户端需先获取服务端 RSA 公钥，用公钥加密密码后再发送；服务端用对应私钥解密后再做 bcrypt 哈希处理。

GET `/api/v1/auth/public-key`

无需认证，可公开访问。

响应：

```
{
  "code": 0,
  "data": {
    "key_id": "rsa_key_20240101_001",
    "public_key": "-----BEGIN PUBLIC KEY-----\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA...\n-----END PUBLIC KEY-----",
    "algorithm": "RSA/ECB/PKCS1Padding",
    "expires_at": "2024-01-01T12:30:00Z"
  }
}
```


字段	说明
`key_id`	公钥标识，传密码时服务端据此找到对应
`public_key`	PEM 格式 RSA 公钥
`algorithm`	加密算法：`RSA/ECB`
`expires_at`	公钥有效期 30 分钟过期需重新获取

前端加密示例（JavaScript）：

```
import JSEncrypt from 'jsencrypt'

const encryptPassword = (plainPassword, publicKey) => {
  const encrypt = new JSEncrypt()
  encrypt.setPublicKey(publicKey)
  return encrypt.encrypt(plainPassword) // 返回 Base64 字符串
}

// 使用示例
const { key_id, public_key } = await fetch('/api/v1/auth/public-key').then(r => r.json()).then(r => r.data)
const encryptedPassword = encryptPassword('Aa123456!', public_key)
// 登录时传入 key_id + encryptedPassword
```

账号密码登录

POST `/api/v1/auth/login`

> 密码安全： `password` 字段必须使用 RSA 公钥加密后传输，不得明文传输。请先调用 `GET /auth/public-key` 获取公钥。

```
{
  "email": "user@example.com",
  "key_id": "rsa_key_20240101_001",
  "password": "Base64(RSA_ENCRYPT('Aa123456!', public_key))"
}
```

字段	类型	必填	说明
`email`	string	是	登录邮
`key_id`	string	是	公钥标 `GET /auth/ key` 获
`password`	string	是	RSA 加 Base6 字符串

响应：

```
{
  "code": 0,
  "data": {
    "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "refresh_token": "rt_01HX...",
    "token_type": "Bearer",
    "expires_in": 7200,
    "user": {
      "user_id": "u_01HX...",
      "email": "user@example.com",
      "user_type": "personal",
      "membership_level": "basic"
    }
  }
}
```


HTTP 状态码： `200` 成功， `401` 账号或密码错误， `400` key_id 无效或已过期， `1102` 账号已冻结

手机号验证码登录

POST `/api/v1/auth/login/phone`

```
{
  "phone": "13800138000",
  "sms_code": "123456"
}
```

登出

POST `/api/v1/auth/logout`

将当前 `access_token` 加入 Redis 黑名单，立即失效。

响应： `{"code": 0, "message": "success"}`

刷新 Token

POST `/api/v1/auth/refresh`

```
{ "refresh_token": "rt_01HX..."
}
```

响应：

```
{
  "code": 0,
  "data": {
```

```
    "access_token": "eyJhbGc
i...(新)",
    "expires_in": 7200
  }
}
```

找回密码

POST `/api/v1/auth/forgot-password`

```
{ "email": "user@example.com" }
```

向邮箱发送重置链接，链接有效期 30 分钟。

POST `/api/v1/auth/reset-password`

> 密码安全： `new_password` 必须使用 RSA 公钥加密后传输。

```
{
  "token": "reset_token_from_em
ail",
  "key_id": "rsa_key_20240101_0
01",
  "new_password": "Base64(RSA_E
```

```
NCRYPT('Bb654321!', public_key))"
}
```

查询当前用户权限

GET `/api/v1/auth/me/permissions`

```
{
  "code": 0,
  "data": {
    "roles": ["user", "enterprise_admin"],
    "permissions": ["chat:use", "model:list", "billing:view", "member:manage"]
  }
}
```

2.2 短信验证码

发送短信验证码

POST `/api/v1/common/sms/send`

```
{
  "phone": "13800138000",
  "purpose": "login"
}
```

`purpose` 值	说明
`login`	登录验证
`register`	注册验证
`reset_password`	找回密码
`change_phone`	更换手机号

频率限制： 同手机号 60 秒内只能发送 1 次，每日最多 10 次。

验证短信验证码

POST `/api/v1/common/sms/verify`

```
{
  "phone": "13800138000",
  "code": "123456",
  "purpose": "login"
}
```

2.3 第三方平台 OAuth 登录

支持个人用户和企业用户分别接入不同第三方平台，统一走 OAuth 2.0 / OIDC 协议。

获取授权跳转 URL

GET `/api/v1/auth/oauth/{provider}/authorize`

查询参数	必填	说明
`redirect_uri`	是	回调地址
`state`	是	防 CSRF 随机串

响应：

```
{
  "code": 0,
  "data": {
    "authorize_url": "https://open.weixin.qq.com/connect/qrconnect?..."
```

```
  }
}
```

处理授权回调

POST `/api/v1/auth/oauth/{provider}/callback`

```
{
  "code": "auth_code_from_provider",
  "state": "state_string"
}
```

响应（首次登录自动注册）：

```
{
  "code": 0,
  "data": {
    "is_new_user": true,
    "access_token": "eyJhbGci...",
    "refresh_token": "rt_01HX...",
    "expires_in": 7200,
    "user": {
      "user_id": "u_01HX...",
```

```
      "nickname": "张三",
      "avatar": "https://...",
      "bound_provider": "wecha
t"
    }
  }
}
```

移动端登录

POST `/api/v1/auth/oauth/{provider}/mo
bile`

```
{
  "code": "js_code_from_wechat_
mp"
}
```

适用于微信小程序 `wx.login()` 返回的 `code` ，直接换取平台 Token。

支持的第三方平台（provider）

provider	平台	适用
`wechat`	微信	个人

provider	平台	适用
	（扫码登录）	
`wechat_mp`	微信小程序	个人
`alipay`	支付宝	个人
`github`	GitHub	个人/企业
`google`	Google	个人/企业
`dingtalk`	钉钉	企业
`feishu`	飞书	企业
`wecom`	企业微信	企业
`ldap`	LDAP / AD	企业
`saml`	SAML 2.0	企业

2.4 第三方账号绑定与解绑

查询已绑定列表

GET `/api/v1/auth/bindings`

```
{
  "code": 0,
  "data": {
    "bindings": [
      {
        "provider": "wechat",
        "provider_user_id": "oX
XXxx...",
        "nickname": "微信昵称",
        "avatar": "http
s://...",
        "bound_at": "2024-01-01
T00:00:00Z"
      },
      {
        "provider": "github",
        "provider_user_id": "12
345678",
        "nickname": "user123",
        "bound_at": "2024-02-01
T00:00:00Z"
      }
    ]
  }
}
```


绑定第三方账号

POST `/api/v1/auth/bindings/{provider}`

```
{
  "code": "auth_code_from_provider",
  "state": "state_string"
}
```

解绑第三方账号

DELETE `/api/v1/auth/bindings/{provider}`

注意： 若账号仅通过此第三方登录且未设置密码，解绑前需先设置密码，否则返回 `400` 。

2.5 企业 SSO 配置

查询 SSO 配置

GET `/api/v1/enterprise/sso`

创建 SSO 配置

POST `/api/v1/enterprise/sso`

```
{
  "type": "saml",
  "idp_metadata_url": "https://idp.company.com/metadata.xml",
  "attribute_mapping": {
    "email": "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress",
    "name": "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name"
  }
}
```

`type` 枚举：`saml` / `oidc` / `ldap`

更新 SSO 配置

PUT `/api/v1/enterprise/sso`

删除 SSO 配置

DELETE `/api/v1/enterprise/sso`

测试 SSO 连通性

POST `/api/v1/enterprise/sso/test`

```
{
  "code": 0,
  "data": {
    "status": "success",
    "latency_ms": 128,
    "user_count": 150
  }
}
```

2.6 RBAC 角色权限管理

角色列表

GET `/admin/v1/roles`

```
{
  "code": 0,
  "data": {
    "items": [
      {
        "role_id": "role_admi
```

```
n",
        "name": "超级管理员",
        "description": "拥有所
有权限",
        "is_system": true,
        "permission_count": 48,
        "user_count": 3
      }
    ]
  }
}
```

创建角色

POST `/admin/v1/roles`

```
{
  "name": "财务审核员",
  "description": "可查看账务和
审核开票申请"
}
```

更新角色

PUT `/admin/v1/roles/{role_id}`

删除角色

DELETE `/admin/v1/roles/{role_id}`

注意： 系统内置角色（`is_system: true`）不可删除。

查询角色权限

GET `/admin/v1/roles/{role_id}/permissions`

```
{
  "code": 0,
  "data": {
    "permissions": [
      "accounting:view",
      "accounting:reconcile",
      "invoice:process"
    ]
  }
}
```

配置角色权限

PUT `/admin/v1/roles/{role_id}/permissions`

```
{
  "permissions": ["accounting:view", "invoice:process"]
}
```

查询用户角色

GET `/admin/v1/users/{user_id}/roles`

分配用户角色

PUT `/admin/v1/users/{user_id}/roles`

```
{ "roles": ["role_finance_reviewer"] }
```

2.7 管理端登录（独立认证流程）

步骤一：账号密码登录

POST `/admin/v1/auth/login`

```
{
  "username": "admin@example.com",
  "password": "AdminPass123!"
}
```

响应：

```
{
  "code": 0,
  "data": {
    "temp_token": "tmp_01HX...",
    "expires_in": 300,
    "require_2fa": true,
    "2fa_type": "totp"
  }
}
```

步骤二：2FA 验证

POST `/admin/v1/auth/2fa/verify`

```
{
  "temp_token": "tmp_01HX...",
  "code": "123456"
}
```

响应： 返回正式 `access_token` （8 小时有效，IP 绑定）

管理端登出

POST `/admin/v1/auth/logout`

管理端 Token 刷新

POST `/admin/v1/auth/refresh`

> 管理端 Token 无法静默刷新，过期后需重新完成完整登录流程（含 2FA）。

01-user-api.md

用户模块

用户分为两种类型，注册时通过 `user_type` 字段区分：

- 个人用户（personal）：个人开发者，手机号/邮箱注册，个人实名认证
- 企业用户（enterprise）：企业主账号，企业认证后可管理子成员账号

```
企业账号（enterprise）
  └──── 成员账号（member）──── 由企业管理员邀请，共享企业钱包与配额
```

1.1 通用用户接口

注册

POST `/api/v1/users/register`

> 密码安全：`password` 字段必须使用 RSA 公钥加密后传输，不得明文传输。请先调用 `GET /api/v1/auth/public-key` 获取公钥，详见授权模块 2.1。

请求体：

```
{
  "user_type": "personal",
  "email": "user@example.com",
  "key_id": "rsa_key_20240101_001",
  "password": "Base64(RSA_ENCRYPT('Aa123456!', public_key))",
  "phone": "13800138000",
  "captcha_token": "captcha_xxx"
}
```

字段	类型	必填
`user_type`	string	是
`email`	string	是

字段	类型	必填
`key_id`	string	是
`password`	string	是
`phone`	string	否
`enterprise_name`	string	企业必
`captcha_token`	string	是

响应：

```
{
  "code": 0,
  "data": {
    "user_id": "u_01HX...",
    "email": "user@example.com",
    "user_type": "personal",
```

```
    "created_at": "2024-01-01T12:00:00Z"
  }
}
```

HTTP 状态码： `201` 创建成功， `409` 邮箱已注册

查询当前用户信息

GET `/api/v1/users/me`

响应：

```
{
  "code": 0,
  "data": {
    "user_id": "u_01HX...",
    "email": "user@example.com",
    "phone": "138****8000",
    "user_type": "personal",
    "nickname": "张三",
    "avatar": "https://oss.example.com/avatars/xxx.jpg",
    "realname_verified": true,
    "membership_level": "basic",
```

```
    "membership_expires_at": "2024-12-31T23:59:59Z",
    "created_at": "2024-01-01T00:00:00Z"
  }
}
```

更新个人信息

PUT `/api/v1/users/me`

```
{
  "nickname": "李四",
  "avatar": "https://oss.example.com/avatars/new.jpg"
}
```

修改密码

PUT `/api/v1/users/me/password`

> 密码安全：`old_password` 和 `new_password` 均必须使用 RSA 公钥加密后传输，不得明文传输。

```
{
  "key_id": "rsa_key_20240101_001",
  "old_password": "Base64(RSA_ENCRYPT('Aa123456!', public_key))",
  "new_password": "Base64(RSA_ENCRYPT('Bb654321!', public_key))"
}
```

字段	类型	必填	说
`key_id`	string	是	公 `G` /a ke
`old_password`	string	是	当 RS Ba 字
`new_password`	string	是	新 加 字

HTTP 状态码： `200` 成功， `400` 旧密码错误或 key_id 无效/已过期

发送邮箱验证码

POST `/api/v1/users/me/verify/email`

```
{
  "email": "user@example.com",
  "purpose": "register"
}
```

`purpose` 枚举：`register` / `reset_password` / `change_email`

管理员查询用户

GET `/admin/v1/users/{user_id}`

GET `/admin/v1/users` — 用户列表

查询参数	类型	说明
`user_type`	string	`personal` / `enterprise`
`status`	string	`active` / `frozen`

查询参数	类型	说明
`keyword`	string	搜索邮箱/手机号/昵称
`page`	int	页码
`page_size`	int	每页条数

1.2 API Key 管理

查询 API Key 列表

GET `/api/v1/users/me/apikeys`

查询参数	说明
`key_type`	`production` / `test`
`status`	`active` / `disabled` / `expired`

响应：

```
{
  "code": 0,
  "data": {
    "items": [
      {
        "key_id": "key_01HX...",
        "name": "生产环境主 Key",
        "key_prefix": "sk-prod-abc1****",
        "key_type": "production",
        "status": "active",
        "last_used_at": "2024-01-01T12:00:00Z",
        "expires_at": null,
        "created_at": "2024-01-01T00:00:00Z"
      }
    ],
    "total": 3
  }
}
```

创建 API Key

POST `/api/v1/users/me/apikeys`

```
{
  "name": "生产环境主 Key",
  "key_type": "production",
  "expires_at": null,
  "allowed_models": [],
  "allowed_ips": []
}
```

字段	类型	说明
`name`	string	Key 便于
`key_type`	string	`pr` `te`
`expires_at`	string|null	过期 表示
`allowed_models`	array	允许 ID 为 空数
`allowed_ips`	array	IP 白 空数

响应（仅在创建时返回完整 Key，之后不再显示）：

```
{
  "code": 0,
  "data": {
    "key_id": "key_01HX...",
    "api_key": "sk-prod-abcdefg
hijklmnopqrstuvwxyz123456",
    "name": "生产环境主 Key",
    "key_type": "production"
  }
}
```

删除 API Key

DELETE `/api/v1/users/me/apikeys/{key_id}`

HTTP 状态码： `200` 成功， `404` Key 不存在

1.3 个人实名认证

提交实名认证

POST

`/api/v1/users/me/verify/realname`

```
{
  "real_name": "张三",
  "id_card": "110101199001011234",
  "id_card_front": "https://oss.example.com/id_cards/front.jpg",
  "id_card_back": "https://oss.example.com/id_cards/back.jpg"
}
```

查询实名认证状态

GET

`/api/v1/users/me/verify/realname`

```
{
  "code": 0,
  "data": {
    "status": "verified",
    "real_name": "张**",
    "verified_at": "2024-01-01T12:00:00Z"
```

```
  }
}
```

`status` 枚举：`unverified` / `pending` / `verified` / `rejected`

1.4 企业用户认证与信息

提交企业认证

POST `/api/v1/users/me/verify/enterprise`

```
{
  "enterprise_name": "XX 科技有限公司",
  "credit_code": "91110000XXXXXXXX",
  "license_image": "https://oss.example.com/licenses/xxx.jpg",
  "contact_name": "李四",
  "contact_phone": "1390013900
```

```
0"
}
```

查询企业信息

GET `/api/v1/enterprise/profile`

```
{
  "code": 0,
  "data": {
    "enterprise_id": "ent_01HX...",
    "enterprise_name": "XX 科技有限公司",
    "credit_code": "9111****",
    "verify_status": "verified",
    "member_count": 12,
    "contact_name": "李四",
    "contact_phone": "139****9000"
  }
}
```

更新企业信息

PUT `/api/v1/enterprise/profile`

1.5 企业成员管理

成员列表

GET `/api/v1/enterprise/members`

```
{
  "code": 0,
  "data": {
    "items": [
      {
        "member_id": "u_01HX...",
        "email": "dev@company.com",
        "nickname": "开发同学",
        "role": "member",
        "status": "active",
        "quota": {
          "tokens_per_day": 500000,
          "tokens_used_today": 12000
        },
        "joined_at": "2024-01-01T00:00:00Z"
      }
    ],
```

```
    "total": 12
  }
}
```

邀请成员

POST `/api/v1/enterprise/members/invite`

```
{
  "email": "newmember@company.com",
  "role": "member"
}
```

移除成员

DELETE `/api/v1/enterprise/members/{member_id}`

查询成员配额

GET `/api/v1/enterprise/members/{member_id}/quota`

设置成员配额

PUT `/api/v1/enterprise/members/{member_id}/quota`

```
{
  "tokens_per_day": 1000000,
  "requests_per_day": 500
}
```

修改成员角色

PUT `/api/v1/enterprise/members/{member_id}/role`

```
{ "role": "admin" }
```

`role` 枚举：`admin` / `member`

00-overview.md

通用规范

本文档定义平台所有 API 接口的基础规范，所有模块均遵循此约定。

---

1. Base URL

端	Base URL
用户控制台	`https://api.example`
平台管理端	`https://api.example`

---

2. 认证方式

2.1 JWT Token（用户界面操作）

登录成功后获得 `access_token` ，携带于请求头：

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

- 有效期：2 小时（用户端）/ 8 小时（管理端，IP 绑定）
- 过期后调用 `POST /auth/refresh` 换取新 Token

2.2 API Key（程序调用）

用于程序直接调用模型接口，格式为 `sk-` 前缀：

```
Authorization: Bearer sk-prod-xxxxxxxxxxxxxxxxxxxxxxxx
```

- 在控制台「API Key 管理」页创建
- 支持按类型（production / test）、IP 白名单、模型白名单和过期时间分类管理

2.3 管理端双因素认证（2FA）

管理员登录需完成两步：

```
步骤 1：POST /admin/v1/auth/login      → 返回临时 token（5 分钟有效）
步骤 2：POST /admin/v1/auth/2fa/verify  → 验证 TOTP/短信，返回正式 access_token
```

2.4 密码传输安全（RSA 加密）

所有涉及密码的接口（注册、登录、修改密码、重置密码）均不得明文传输密码。

```
步骤 1：GET /api/v1/auth/public-key   → 获取 RSA 公钥（有效期 30 分钟）
步骤 2：前端使用公钥加密密码          → Base64(RSA_ENCRYPT(password, public_key))
步骤 3：请求时携带 key_id + 加密密文   → 服务端用私钥解密后再做 bcrypt 处理
```

详见 02-auth-api.md。

---

3. 统一响应格式

所有接口均返回以下结构：

```
{
  "code": 0,
  "message": "success",
  "data": {},
  "request_id": "req_550e8400-e29b-41d4-a716-446655440000",
  "timestamp": 1704067200
}
```

字段	类型	说
`code`	int	0
`message`	string	人
`data`	object/array/null	业
`request_id`	string	唯
`timestamp`	int	时

3.1 分页响应格式

列表接口统一返回分页结构：

```
{
  "code": 0,
  "message": "success",
  "data": {
    "items": [],
    "total": 100,
    "page": 1,
    "page_size": 20,
    "total_pages": 5
  }
}
```

3.2 通用分页查询参数

参数	类型	默认值
`page`	int	1
`page_size`	int	20
`order_by`	string	`created_at`

参数	类型	默认值
`order`	string	`desc`

---

4. HTTP 状态码

状态码	含义
`200`	成功
`201`	创建成功
`400`	请求参数错误
`401`	未认证（Token 缺失或无效）
`403`	无权限
`404`	资源不存在
`409`	资源冲突（如邮箱已注册）
`422`	参数格式校验失败（Pydantic 错误）
`429`	请求频率超限

状态码	含义
`500`	服务器内部错误
`503`	服务暂不可用（维护中）

---

5. 错误码体系

范围	分类	常见示例
`1000-1099`	认证错误	`1001` Toker 已过期，`100` 无效，`1003`
`1100-1199`	授权错误	`1101` 无权限 账号已冻结， 验证失败
`2000-2099`	用户错误	`2001` 邮箱已注册， 手机号已注册 用户不存在
`2100-2199`	实名认证	`2101` 已实名 认证审核中， 企业认证失败

范围	分类	常见示例
`3000-3099`	支付错误	`3001` 余额不足，`3` 订单不存在，订单已支付
`3100-3199`	代金券错误	`3101` 券码无效，`3` 券已使用，`3` 券已过期
`4000-4099`	模型错误	`4001` 模型不存在，模型已禁用，提供商不可用
`4100-4199`	限流错误	`4101` 超出限制，`4102` 限制
`4200-4299`	队列错误	`4201` 队列已满，`4` 请求已取消
`5000-5099`	网关错误	`5001` API K 无效，`5002` 已过期，`500` 被封禁

范围	分类	常见示例
`6000-6099`	计费错误	`6001` 计费规则不存 账单生成失败
`6100-6199`	账务错误	`6101` 对账失败， 发票申请失败

错误响应示例

```
{
  "code": 1001,
  "message": "Token 已过期，请重新登录",
  "data": null,
  "request_id": "req_550e8400-e29b-41d4-a716-446655440000",
  "timestamp": 1704067200
}
```

---

6. 请求头规范

请求头	必填
`Authorization`	是（除公开接口）
`Content-Type`	POST/PUT 时必填
`Accept-Language`	否
`X-Request-ID`	否
`X-Client-Version`	否

---

7. 流式响应（SSE）

模型对话接口支持 Server-Sent Events 流式输出：

```
POST /api/v1/chat/completions
Content-Type: application/json
Accept: text/event-stream
```

```
{"model": "gpt-4o", "messages": [...], "stream": true}
```

响应格式：

```
data: {"id":"chatcmpl-001","choices":[{"delta":{"content":"你好"},"index":0}]}

data: {"id":"chatcmpl-001","choices":[{"delta":{"content":"！"},"index":0}]}

data: [DONE]
```

---

8. 时间与数据类型约定

类型	约定
时间格式	ISO 8601，统一 UTC：`2`
时间戳	Unix 秒级整数
金额	字符串 Decimal，保留 2
Token 数量	integer

类型	约定
枚举值	小写下划线：`"user_reg`
ID 字段	字符串， 带业务前缀：`"u_01HX..`
布尔值	`true` / `false`

00e-baofoo-sms-api.md

宝付短信渠道接口

本文档描述平台短信发送所使用的宝付（Mandao）短信网关接口规范，供后端通知服务（Celery Worker）集成使用。

环境地址

环境	接口地址
测试环境	`http://10.254.90.70`
准生产环境	`https://qas-sms.mandao.com/sendS`
生产环境	`https://sms.mandao.`

> 注意： 生产环境需在宝付侧开通服务器出口 IP 白名单，部署前需提前申请。

请求规范

- 请求方式： `POST`
- Content-Type： `multipart/form-data`

请求参数

参数名	类型	必填	说
`userName`	string	是	账
`userPass`	string	是	密
`mobile`	string	是	手 多 或
`content`	string	是	短
`domain`	string	否	应 建 不
`singCode`	string	否	签
`regTime`	string	否	请

参数名	类型	必填	说
`sendType`	int	否	发 3
`sms_business`	string	否	短
`memberId`	string	否	商
`businessType`	int	否	系 (
`areaCode`	string	否	地
`senderId`	string	否	发
`requestId`	string	否	请 用 建

sendType 枚举

值	说明
`1`	注册发短信验证码
`2`	注册成功发送短信
`3`	直接发送短信内容（默认）

businessType 枚举

值	说明
`1`	签约发送短信
`2`	支付成功通知
`3`	预支付发送短信
`4`	普通业务通知

响应参数

字段	类型	说明
`success`	bool	请求是否
`result`	int	发送结果 `0` 表示发送
`errorCode`	string|null	错误码， 成功时为 `null`

字段	类型	说明
`errorMsg`	string|null	错误描述 成功时为 `null`

判断标准： `success == true` 且 `result > 0` 为短信发送成功。

成功响应示例：

```
{
  "success": true,
  "result": 1,
  "errorCode": null,
  "errorMsg": null
}
```

失败响应示例：

```
{
  "success": false,
  "result": 0,
  "errorCode": "SMS_001",
  "errorMsg": "手机号格式不正确"
}
```

请求示例

Python（Celery Worker 集成）

```
import httpx
import uuid
from app.config import settings
# 从 KMS/配置中心读取

async def send_sms(mobile: str, content: str, business_type: int = 4) -> bool:
    """
    发送短信
    :param mobile: 手机号，多个用逗号分隔
    :param content: 短信内容
    :param business_type: 业务类型，默认 4（普通业务通知）
    :return: 是否发送成功
    """
    payload = {
        "userName": settings.BAOFOO_SMS_USERNAME,
        "userPass": settings.BAOFOO_SMS_PASSWORD,
        "mobile": mobile,
```

```
        "content": content,
        "domain": settings.APP_
NAME,
        "sendType": "3",
        "businessType": str(bus
iness_type),
        "requestId": str(uuid.u
uid4()),
    }

    async with httpx.AsyncClien
t(timeout=10.0) as client:
        response = await clien
t.post(settings.BAOFOO_SMS_URL,
data=payload)
        result = response.json
()

    success = result.get("succe
ss") is True and (result.get("r
esult") or 0) > 0
    if not success:
        logger.error(
            "宝付短信发送失败",
            mobile=mobile,
            error_code=result.g
et("errorCode"),
            error_msg=result.ge
t("errorMsg"),
```

```
        )
    return success
```

配置项（存入 KMS / 环境变量）

配置键	说明
`BAOFOO_SMS_USERNAME`	账号（userNan
`BAOFOO_SMS_PASSWORD`	密码（userPas
`BAOFOO_SMS_URL`	当前环境接
`APP_NAME`	应用名称， domain 字

平台短信场景映射

场景	sendType	busines
登录验证码	`1`	`4`

场景	sendType	busines
注册验证码	`1`	`4`
找回密码	`3`	`4`
充值成功	`3`	`2`
余额不足	`3`	`4`
会员到期提醒	`3`	`4`

注意事项

1. 生产环境 IP 白名单： 部署前需将服务器出口 IP 提交给宝付侧开通，否则生产环境请求会被拦截。

2. requestId 幂等： 建议每次请求生成唯一 UUID 作为 `requestId`，便于宝付侧排查问题。
3. 凭证安全： `userName` 和 `userPass` 禁止明文写入代码或 Git，统一存入 KMS，参见 00d-preparation-checklist.md。
4. 发送频率： 同一手机号短信发送需在业务层做频率限制（如 60 秒 1 次），避免触发运营商拦截。
5. 境外手机号： 发送境外号码时需填写 `areaCode`（如中国大陆 `+86`、香港 `+852`）。

00d-preparation-checklist.md

上线前准备清单

本文档列出平台上线前需要申请、注册或准备的所有外部资源，涵盖域名、资质、第三方服务凭证和基础设施。

> 建议提前 4-8 周 开始申请，部分审核（ICP 备案、支付商户号）耗时较长。

一、域名与备案

资源	用途
主域名（如 `example.com`）	品牌域名
`console.example.com`	用户控制台
`admin.example.com`	平台管理端

资源	用途
`api.example.com`	后端 API 入
ICP 备案	中国大陆用
公安网安备案	ICP 备案后 天内完成
SSL 证书	HTTPS 加密

二、企业资质（前置条件）

> 以下为支付、短信、发票等服务的申请前提，优先准备。

资质	用途
营业执照（企业）	几乎所有第三方
法人身份证正反面	支付商户号、短
对公银行账户	支付宝/ 微信支付商户结

资质	用途
银行开户许可证	微信支付商户号
行业资质（如有）	部分行业有特殊

三、第三方登录（OAuth）

3.1 个人用户登录渠道

平台	需申请内容	申请入
微信（扫码登录）	AppID + AppSecret	微信开 open.w → 网站
微信小程序	AppID + AppSecret	微信公 mp.we → 小程
支付宝	APPID + 应用私钥 + 支付宝公钥	支付宝 open.a
GitHub	Client ID + Client Secret	GitHub → Dev

平台	需申请内容	申请入
		setting Apps
Google	Client ID + Client Secret	Google Conso Servic Crede

3.2 企业用户登录渠道

平台	需申请内容	申请入口
钉钉	AppKey + AppSecret	钉钉开放平 open.dingt → 应用开发
飞书	App ID + App Secret	飞书开放平 open.feish 开发者后台
企业微信	CorpID + AgentID + Secret	企业微信管 work.weixin

> 注意： 微信扫码登录需要已上线的网站，审核时需填写网站 URL，

> 建议域名备案完成后再申请。

四、支付渠道

渠道	需申请内容	申请入口
支付宝支付	APPID + 商户私钥 + 支付宝公钥（或证书）	支付宝开 → 支付
微信支付	商户号（MchID）+ API v3 密钥 + 商户证书（.p12）	pay.weix → 商户
宝付（Baofoo）	商户号 + API 密钥 + 证书	baofoo.c 商户合作

> 重要： 支付宝和微信支付需要企业营业执照 + 对公银行账户，个人无法申请。微信支付还需提交行业材料。

五、AI 模型提供商

提供商	需申请内容	申请
OpenAI	API Key	plat API
阿里云百炼	API Key （DashScope）	bail
Anthropic （Claude）	API Key	cons
Google （Gemini）	API Key	aist Goo
百度文心	API Key + Secret Key	qian
智谱 AI （GLM）	API Key	ope

> 注意： OpenAI 和 Anthropic 在国内访问需要代理；阿里云百炼支持国内直连，建议作为备用或主力提供商。

六、短信服务

资源	说明
阿里云短信（推荐）	AccessKey ID + AccessKey Secret
短信签名	如「【XX平台】」，需营业执照，审核约 1-2 个工作日
短信模板	登录验证码、 注册验证码、 找回密码、 余额告警等模板， 各需单独审核

需要申请的短信模板：

- 登录验证码：`您的登录验证码为 ${code}，30分钟内有效。`
- 注册验证码：`您的注册验证码为 ${code}，30分钟内有效。`
- 找回密码：`您的重置密码验证码为 ${code}，30分钟内有效。`

- 余额不足：`您的账户余额仅剩 ${balance} 元，请及时充值。`
- 会员到期：`您的${plan}会员将于 ${date} 到期，请及时续费。`

七、邮件服务

资源	说明
阿里云邮件推送（推荐）	AccessKey ID + Ac
发信域名	如 `mail.example` 记录
发信地址	如 noreply@example.c 需在控制台验证

需要配置的邮件模板：

- 邮箱注册验证
- 找回密码重置链接
- 企业成员邀请
- 月度账单通知

- 充值成功确认
- 会员到期提醒

八、云存储

资源	用途	访
阿里云 OSS	文件存储	仓 获 + A
Bucket 名称	用于存储用户头像、营业执照、账单 PDF	建 B 可
CDN （可选）	静态资源加速	绑 B 配

九、基础设施（云服务器）

资源	推荐规格
云服务器 ECS	4 核 8G × 2（业务后端 + AI 网关）
RDS PostgreSQL	2 核 4G，100GB 存储
Redis	1G 标准版
ClickHouse	自建（4 核 8G）或云服务
Kafka	自建或阿里云消息队列 Kafka 版
负载均衡 SLB	按量付费
对象存储 OSS	按量付费

十、安全与合规

资源	说明
阿里云 KMS	存储 AK/SK、 支付密钥等敏感配置 避免明文写入代码
WAF（Web 应用防火墙）	防 SQL 注入、XSX、 攻击，管理端尤其重
DDoS 防护	基础版免费， 业务稳定后按需升级
等保备案 （推荐）	对企业客户有一定信 二级等保

十一、微信公众号（可选）

如需通过公众号推送服务通知：

资源	说明
微信公众号 AppID + AppSecret	服务号（有模板消息权 > 订阅号
消息模板	充值成功、余额告警、 会员到期等， 在公众号后台申请

资源	说明
JS-SDK 域名配置	需将前端域名添加到公 JS 接口安全域名

十二、准备时间线参考

```
第 1-2 周：
  □ 注册域名
  □ 提交 ICP 备案
  □ 准备企业资质材料
  □ 申请 OpenAI / 阿里云百炼 API Key（立即可用）

第 2-4 周：
  □ ICP 备案审核中
  □ 申请支付宝支付商户号
  □ 申请微信支付商户号
  □ 申请阿里云短信签名 + 模板
  □ 配置邮件发信域名 + 模板
  □ 申请微信开放平台网站应用（需域名已备案）

第 4-6 周：
  □ ICP 备案完成 → 公安网安备案
```

```
  □ 微信支付审核完成
  □ 申请钉钉 / 飞书 / 企业微信应用
  □ 购买云服务器 + 数据库 + Redis
  □ 配置 SSL 证书 + CDN

第 6-8 周：
  □ 所有资质到位，进行联调测试
  □ 完成等保备案（可选）
  □ 开始公测
```

十三、配置汇总表（上线前填写）

> 建议将以下信息存入 阿里云 KMS 或 Vault，不要明文写入代码或 `.env` 提交到 Git。

分类	配置项	示
域名	主域名	
域名	ICP 备案号	
微信登录	AppID	
微信登录	AppSecret	

分类	配置项	
微信小程序	AppID	
微信小程序	AppSecret	
微信支付	商户号 MchID	
微信支付	API v3 密钥	
微信支付	商户证书序列号	
支付宝登录/支付	APPID	
支付宝	应用私钥	
支付宝	支付宝公钥	
宝付	商户号	
宝付	API 密钥	
GitHub OAuth	Client ID	
GitHub OAuth	Client Secret	
Google OAuth	Client ID	
Google OAuth	Client Secret	

分类	配置项	
钉钉	AppKey	
钉钉	AppSecret	
飞书	App ID	
飞书	App Secret	
企业微信	CorpID	
企业微信	AgentID + Secret	
OpenAI	API Key	
阿里云百炼	API Key	
Anthropic	API Key	
百度文心	API Key + Secret Key	
智谱 AI	API Key	
阿里云短信	AccessKey ID	
阿里云短信	AccessKey Secret	
阿里云短信	短信签名	
阿里云邮件	AccessKey ID	
阿里云邮件	发信地址	

分类	配置项	
阿里云 OSS	Bucket 名称	
阿里云 OSS	AccessKey ID	
阿里云 OSS	AccessKey Secret	
数据库	PostgreSQL 连接串	
缓存	Redis 连接串	
RSA 密钥对	公钥（用于密码加密）	
RSA 密钥对	私钥（服务端保存）	
JWT	Secret Key（业务后端）	
JWT	Secret Key（管理端）	

00c-tech-stack.md

技术栈方案（已选定：方案 C 全 Python）

> 已选定方案 C：Python FastAPI 全栈，AI 网关与业务后端统一使用 Python，前端使用 Next.js 14。

架构总览

```
┌──────────────────────────────────────────────────────────────────────────────────────────────────────────────┐
│                         用户控制台
│
│   Next.js 14 + TypeScript + shadcn/ui + TailwindCSS
│
│   Vercel AI SDK（流式对话）+ ECharts（用量仪表盘）
```

```
│
│   NextAuth.js (OAuth) + TanSt
ack Query + Zustand
│
├──────────────────────────────
──────────────────────────────
──────────────────────────────
──────────────────────────────
──────────────┤
│                              平台
管理端
│
│   Next.js 14 + TypeScript + A
nt Design Pro
│
├──────────────────────────────
──────────────────────┬───────
──────────────────────────────
──────────────────────────────
──────────────┤
│    Python 业务后端       │
Python AI 网关
│
│    FastAPI + Celery     │
FastAPI + httpx
│
│    SQLAlchemy + PG      │
openai/anthropic/dashscope SDK
│
```

```
│    支付宝/微信 SDK       │
Redis 优先级队列 + 滑动限流
│
│    JWT + RBAC           │
tenacity 熔断降级
│
├──────────────────────────────
──────────────────┴─────────────
──────────────────────────────
──────────────────────────────
──────────────────────────────
──────────────┤
│                                存
储层
│
│   PostgreSQL 16（主业务）   Re
dis 7（缓存/队列/限流）
│
│   ClickHouse 24（Token 用量分
析）   OSS（文件存储）
│
├──────────────────────────────
──────────────────────────────
──────────────────────────────
──────────────────────────────
──────────────┤
│                                消息
与任务
│
│   Kafka（服务间事件）   Celery
```

一、用户控制台前端

1.1 框架与工程化

技术	版本	用途
Next.js	14（App Router）	框架，SSR/SSG 内置路由友好
TypeScript	5.x	全量类型安
TailwindCSS	3.x	原子化 CS 快速构建
shadcn/ui	latest	基于 Radix TailwindCS 高度可定制
Ant Design	5.x	补充复杂业（Table、Upload）
Turbopack	内置	Next.js 默认打包器极速热更新
pnpm	8.x	包管理器

1.2 状态管理

技术	用途
Zustand	全局客户端状态（用户信息、会员等级、未读消息数、主题）
TanStack Query v5	服务端状态（接口缓存、自动重试、乐观更新、后台刷新）
Jotai	局部原子状态（对话页消息列表、设置面板）

1.3 AI 对话与流式输出

技术	用途
Vercel AI SDK	封装 SSE 流式对话，`useCha` hook 对接 `/chat/completion`
`EventSource` API	消息中心实时推送 `GET /messages/stream`

技术	用途
`react-markdown`	渲染 AI 返回 Markdown 内容
`rehype-highlight`	代码块语法高亮（自动识别 50+ 语言）
`remark-gfm`	GitHub Flavored Markdown（表格、任务列表）
KaTeX	数学公式渲染（`$...$` 行内 / `$$...$$` 块级）

1.4 数据可视化

技术	用途
ECharts 5 + `echarts-for-react`	Token 用量趋势折线、T 柱状图、热力图
Recharts	简单图表 （会员套餐用量进度弧形

1.5 表单与校验

技术	用途
React Hook Form	高性能表单（注册/登录/企业认证/充值）
Zod	Schema 校验，前后端共享类型定义

1.6 认证与安全

技术	用途
NextAuth.js v5	OAuth 登录（微信/GitHub/Google JWT session
`jose`	客户端 JWT 解析（读取用户角色、会员等级）
CSRF 保护	Next.js middleware + `next-csrf`

1.7 支付集成（前端）

技术	用途
支付宝 JSSDK	H5 跳转支付
微信 JS-SDK	微信内网页支付
自定义 QR 组件	展示二维码 + 轮询支付状态（`useInterval`）

1.8 UI 交互与动效

技术	用途
Framer Motion	页面切换、卡片入场、流式字符打字动画
`sonner`	Toast 通知（轻量优雅）
`cmdk`	全局命令面板（⌘K，快速切换模型/跳转页面）
Lucide React	图标库（统一风格）

1.9 国际化与其他

技术	用途
`next-intl`	i18n 中英文切换，URL 路由级别
Vitest	单元测试（工具函数、Zod schema）
React Testing Library	组件交互测试
Playwright	E2E 测试（登录/充值/对话完整流程）

1.10 关键页面设计说明

主页（Landing Page / 首页）：

- 未登录态：品牌介绍 + 核心功能亮点卡片 + 定价套餐对比 + 注册/登录 CTA 按钮，SSG 静态生成，SEO 友好

- 已登录态：自动跳转控制台 Dashboard，展示今日用量摘要、余额状态、最近对话入口、系统公告横幅
- 导航栏：Logo + 功能导航 + 登录/注册按钮（未登录）/ 用户头像菜单 + 消息 Badge（已登录）
- 响应式布局，移动端优先，TailwindCSS 断点适配
- Framer Motion 入场动画（Hero 区域文字淡入 + 功能卡片依次滑入）

模型列表页：

- 顶部筛选栏：按提供商（OpenAI / 阿里云 / Anthropic…）、模型类型（对话 / 嵌入 / 图像）、是否支持流式、上下文长度区间多维筛选
- 卡片网格布局：每张模型卡片展示模型名、提供商 Logo、上下文窗口大小、input/output 定价、支持

的能力标签（`Function Calling` / `Vision` / `Code` 等）

- 价格对比入口：点击任意模型卡片右上角「比价」图标，弹出与第三方的价格对比 Sheet（调用 `GET /tokens/price-comparison/{model_id}`）
- 快速发起对话：每个模型卡片有「立即对话」按钮，点击携带 `model_id` 跳转到对话页并预选该模型
- 当前会员等级价格高亮：根据用户会员等级（free / basic / premium）在卡片上显示对应档位的实际价格
- TanStack Query 缓存模型列表（5 分钟 stale time），搜索框使用防抖（300ms）

AI 对话页：

- 左侧历史列表 + 右侧对话区，`useChat` 流式逐字输出

- 消息气泡支持 Markdown + 代码高亮 + KaTeX 公式
- 右下角实时显示本次 Token 消耗与费用
- 流式输出时光标闪烁动画（Framer Motion）

Token 仪表盘：

- 顶部今日/本月用量卡片（数字跳动动效）
- 中间趋势折线图（ECharts，支持时间范围切换）
- 底部 Top 10 模型横向柱状图 + API Key 用量 Table
- TanStack Query 自动后台刷新

充值页：

- 金额快选按钮（50/100/200/500）+ 自定义输入
- 扫码/H5 跳转支付，`useInterval` 轮询支付状态

- 支付成功后 Framer Motion 庆祝动效 + 余额实时刷新

二、平台管理端前端

技术	版本	用途
Next.js	14（App Router）	框架
TypeScript	5.x	
Ant Design Pro	6.x	ProTable / ProForm / ProDescrip
TailwindCSS	3.x	补充自定义
ECharts	5.x	运营数据图
Zustand		全局状态
TanStack Query	v5	接口状态管

三、AI 网关服务（Python）

职责： 模型调用、SSE 流式输出、多提供商路由、优先级队列、限流、降级熔断。

技术	版本	用途
Python	3.12	
FastAPI	0.111+	异步 HTT `Streami`
uvicorn	0.30+	ASGI 服务 + uvicorn
Pydantic v2		请求/响应 提升 5-5
httpx		异步 HTT LLM API
openai SDK	latest	OpenAI /
anthropic SDK	latest	Claude 系
dashscope SDK	latest	阿里云百
qianfan SDK	latest	百度文心

技术	版本	用途
zhipuai SDK	latest	智谱 GLM
redis-py (async)		限流（滑 优先级队
SQLAlchemy 2.0		异步 ORM
tenacity		自动重试 触发降级
prometheus-fastapi-instrumentator		Prometh

优先级队列实现（Redis Sorted Set）：

```
# score = 时间戳 * 1000 - 优先级偏移（P0 偏移最大，最先出队）
priority_offset = {"P0": 4000, "P1": 3000, "P2": 2000, "P3": 1000}
score = time.time() * 1000 - priority_offset[priority]
await redis.zadd(f"model:{model_id}:queue", {request_id: score})
```

```
# 工作协程取出最高优先级
request_id = await redis.zpopmi
n(f"model:{model_id}:queue")
```

四、业务后端服务（Python）

职责： 用户、认证、支付、钱包、账务、会员、营销等业务逻辑。

技术	版本	用途
Python	3.12	
FastAPI	0.111+	HTTP 框 中间件、
uvicorn	0.30+	ASGI 服
Pydantic v2		全链路数
SQLAlchemy 2.0		异步 ORM （`async`
Alembic		数据库版
python-jose		JWT 签发 RS256）

技术	版本	用途
passlib + bcrypt		密码哈希
redis-py (async)		JWT 黑名
Celery	5.x	异步任务 发券/生
celery-beat		定时任务 会员到期
alipay-sdk-python		支付宝支
wechatpayv3		微信支付
python-multipart		文件上传 头像）
oss2 / boto3		阿里云 O
structlog		结构化 J
opentelemetry-sdk		链路追踪 trace_id）
pytest + httpx		单元测试

项目结构（Monorepo 双服务）：

```
backend/
├──── ai_gateway/            # AI 网关服务
│     ├──── main.py
│     ├──── routers/          # chat, models, providers
│     └──── services/         # llm_router, rate_limiter, queue, fallback
│
├──── business/              # 业务后端服务
│     ├──── main.py
│     ├──── routers/          # users, auth, wallet, orders, billing...
│     ├──── services/         # user_service, payment_service...
│     ├──── models/           # SQLAlchemy ORM 模型
│     └──── tasks/            # Celery 异步任务
│
├──── shared/                # 两个服务共享代码
│     ├──── db.py             # 数据库连接池
│     ├──── redis_client.py   #
```

```
Redis 连接
│   ├──── schemas/          # 共享 Pydantic schemas
│   └──── middleware/       # 认证中间件、日志中间件
│
├──── alembic/              # 统一数据库迁移
├──── tests/                # 测试
└──── requirements.txt
```

五、数据存储层

存储	版本	用途
PostgreSQL	16	主业务数据库
Redis	7	缓存 + 队列 + 限流

存储	版本	用途
ClickHouse	24.x	Token 用量分析（写多读少）
阿里云 OSS / MinIO	—	对象存储

ClickHouse token_usage 表结构：

```
CREATE TABLE token_usage (
  request_id    String,
  user_id       String,
  api_key_id    String,
  model_id      String,
  provider_id   String,
  input_tokens  UInt32,
  output_tokens UInt32,
  total_tokens  UInt32,
  input_cost    Decimal(18,8),
  output_cost   Decimal(18,8),
  latency_ms    UInt32,
  created_at    DateTime
```

```
) ENGINE = MergeTree()
PARTITION BY toYYYYMM(created_a
t)
ORDER BY (user_id, model_id, cr
eated_at);
```

六、消息队列（Kafka）

Topic	生产方
`model.request.completed`	AI 网关
`billing.charged`	计费服
`payment.success`	支付回
`user.registered`	业务后
`notification.send`	各业务
`membership.expiring`	Celery

七、基础设施与部署

组件	技术	说
容器化	Docker + docker-compose（开发）	`d` up —一
编排	Kubernetes + Helm（生产）	各 De 按
反向代理	NGINX	SS 静 路
CI/CD	GitHub Actions	lin 构 滚
监控	Prometheus + Grafana	QP P9 队
日志	Loki + Grafana	str 日 全
链路追踪	Jaeger（OpenTelemetry）	浏 业 网

组件	技术	说
告警	Alertmanager + 钉钉 Webhook	错 队 自
密钥管理	阿里云 KMS	AK 加 避

docker-compose 服务清单（开发环境）：

```
services:
  business:       # 业务后端 FastAPI   :8000
  ai_gateway:     # AI 网关 FastAPI    :8001
  celery_worker:  # 异步任务执行
  celery_beat:    # 定时任务调度
  postgres:       # PostgreSQL 16     :5432
  redis:          # Redis 7           :6379
  clickhouse:     # ClickHouse 24     :8123
  kafka:          # Kafka
```

```
:9092
  zookeeper:       # Kafka 依赖
:2181
  nginx:           # 反向代理
:80/:443
```

八、服务拓扑图

```
用户浏览器 / 第三方 API 调用
   └──── NGINX（SSL终止 + 静态资源）
         ├──── /api/v1/chat/* → Python AI 网关（FastAPI，模型调用/SSE流式）
         ├──── /api/v1/* → Python 业务后端（FastAPI，用户/支付/钱包/会员/营销）
         └──── /admin/v1/* → Python 业务后端（管理员接口，RBAC 鉴权）

Python 业务后端
   ├──── PostgreSQL（用户/订单/钱包/会员/营销主业务数据）
   ├──── Redis（JWT黑名单、API Key 缓存）
```

```
  ├──── Celery Worker（发邮件/短信、发券、生成PDF）
  ├──── Celery Beat（月度账单、会员到期检查、对账）
  └──── Kafka（生产：user.registered / payment.success / notification.send）

Python AI 网关
  ├──── Redis（优先级队列 Sorted Set、滑动窗口限流计数）
  ├──── ClickHouse（Token 用量明细批量写入）
  ├──── OpenAI / Anthropic / 阿里云百炼 / 百度 / 智谱...
  └──── Kafka（生产：model.request.completed）
```

00b-portal-guide.md

双端说明：用户控制台 vs 平台管理端

平台包含两个独立的 UI 端，共享同一套后端服务，通过不同的 API 前缀、认证方式和权限体系进行隔离。

---

1. 端点总览

维度	用户控制台
访问对象	注册用户（个人用户 / 企业用户）
前端域名	`https://console.examp`
API 前缀	`/api/v1/`
登录方式	账号密码 / 手机号 / 三方
会话 Token	JWT，有效期 2 小时
数据范围	仅自身账号数据

维度	用户控制台
Token 刷新	支持静默刷新

---

2. 用户控制台 — 功能页面与 API 映射

认证页面

页面	核心 API
登录	`POST /api/v1/auth/l` `/api/v1/auth/login/p`
注册	`POST /api/v1/users/`
找回密码	`POST /api/v1/auth/f` `password` ，`POST /ap` `password`
第三方登录	`GET` `/api/v1/auth/oauth/{`

主控制台页面

页面	核心 API
首页概览（Dashboard）	`GET /api/v1/tokens/us` `/api/v1/wallet/ba` `/api/v1/messages/`
模型广场	`GET /api/v1/mode`
AI 对话	`POST /api/v1/cha` 流式）
对话历史	`GET /api/v1/chat` `/api/v1/chat/hist`
排队状态	`GET /api/v1/chat`

Token 用量统计

页面	核心 API
用量仪表盘	`GET /api/v1/tokens/usage`
每日趋势	`GET /api/v1/tokens/usage`
Top 10 模型	`GET /api/v1/tokens/models`

页面	核心 API
API Key 统计	`GET /api/v1/tokens/apikey`
实时用量	`GET /api/v1/tokens/usage`
价格对比	`GET /api/v1/tokens/comparison`
费用预估	`POST /api/v1/tokens`

财务页面

页面	核心 API
钱包 / 余额	`GET /api/v1/wallet/ba`
充值	`POST /api/v1/wallet/r`
代金券	`GET /api/v1/wallet/voucher` `/api/v1/wallet/voucher`
订单记录	`GET /api/v1/orders`
交易流水	`GET /api/v1/transacti`
月度账单	`GET /api/v1/billing/i`

页面	核心 API
申请开票	`POST /api/v1/wallet/i`

会员与个人

页面	核心 API
会员中心	`GET /api/v1/mem` `/api/v1/membersh`
个人信息	`GET /api/v1/use` `/api/v1/users/me`
企业信息	`GET` `/api/v1/enterpri`
企业成员管理	`GET` `/api/v1/enterpri`
API Key 管理	`GET /api/v1/use`
第三方账号绑定	`GET /api/v1/aut`
安全设置	`PUT /api/v1/use`

消息与通知

页面	核心 API
消息中心	`GET /api/v1/messa` `/api/v1/messages/u`
实时推送	`GET /api/v1/messa` 长连接）
通知偏好设置	`GET` `/api/v1/notificati` `/api/v1/notificati`

---

3. 平台管理端 — 功能页面与 API 映射

管理端登录流程

```
POST /admin/v1/auth/login
→ 验证账号密码，返回临时 token（5分钟）
POST /admin/v1/auth/2fa/verify
→ 验证 TOTP/短信，返回正式 access_token（8h，IP绑定）
POST /admin/v1/auth/logout
```

```
POST /admin/v1/auth/refresh
→ 无法静默刷新，过期需重新登录
```

数据总览

页面	核心 API
运营仪表盘	`GET /admin/v1/stats/over`
收入趋势	`GET /admin/v1/stats/reve`
用户增长	`GET /admin/v1/stats growth`
平台用量 Top 10	`GET /admin/v1/tokens/usa users`

用户管理

页面	核心 API
用户列表	`GET /admin/v1/use` 状态筛选）
用户详情	`GET /admin/v1/use`

页面	核心 API
冻结 / 解冻	`PUT /admin/v1/users/{u` `/admin/v1/users/{u`
重置密码	`PUT /admin/v1/use` `password`
企业列表	`GET /admin/v1/ent`
企业认证审核	`PUT /admin/v1/enterpri` `/admin/v1/enterpri`
角色权限配置	`GET /admin/v1/rol` `/admin/v1/roles/{r`

财务管理

页面	核心 API
充值订单	`GET /admin/v1/ord`
退款审批	`POST /admin/v1/orders/{`
交易流水	`GET /admin/v1/tra`
手动余额调整	`POST /admin/v1/wa`

页面	核心 API
对账管理	`POST /admin/v1/accounti`
开票审核	`GET /admin/v1/acc invoices` ，处理/拒
月度账单	`GET /admin/v1/bil`

模型管理

页面	核心 API
提供商配置	`GET /admin/v1/model providers` ，`POST /admin/v1/model-providers/{id}/test`
模型列表	`GET /admin/v1/model` 禁用
模型定价	`GET /admin/v1/tokens/pri` `/admin/v1/tokens/pri`
限流规则	`GET /admin/v1/model limits`

页面	核心 API
降级策略	`GET /admin/v1/model rules`
队列监控	`GET /admin/v1/models/que`

Token 管理

页面	核心 API
Token 定价配置	`GET /admin/v1/tokens/p`
第三方市场价	`GET /admin/v1/tokens/m prices`
平台用量统计	`GET /admin/v1/tokens/u users`

会员与营销

页面	核心 API
套餐配置	`GET /admin/v1/membersh` `/admin/v1/membersh`
活动管理	`GET /admin/v1/marketin`
券类型配置	`GET /admin/v1/mar types`
批量生成券码	`POST /admin/v1/ma codes/generate`
自动发放规则	`GET /admin/v1/marketin rules`
手动发放	`POST /admin/v1/marketin`

消息与通知

页面	核心 API
公告管理	`GET /admin/v1/messages/ann` 发布/撤回

页面	核心 API
通知模板	`GET /admin/v1/notification`
广播消息	`POST /admin/v1/notification`
发送日志	`GET /admin/v1/notific logs`

系统与网关

页面	核心 API
路由规则	`GET /admin/v1/gateway/ro`
Webhook 配置	`GET /admin/v1/gateway/we`
访问日志	`GET /admin/v1/gateway/lo`
审计日志	`GET /admin/v1/gateway/au logs`
管理员账号	`GET /admin/v1/admins` ,

页面	核心 API
	`/admin/v1/admins`
操作日志	`GET /admin/v1/system/ope logs`

---

4. 管理端专属接口

以下接口仅在 `/admin/v1/` 下存在，对普通用户不可见：

方法	路径
GET	`/admin/v1/stats/overview`
GET	`/admin/v1/stats/revenue`
GET	`/admin/v1/stats/user-grow`
PUT	`/admin/v1/users/{user_id}`
PUT	`/admin/v1/users/{user_id}`
PUT	`/admin/v1/users/{user_id}` `password`
PUT	`/admin/v1/enterprises/{id`

方法	路径
PUT	`/admin/v1/enterprises/{id`
POST	`/admin/v1/wallet/adjust`
GET	`/admin/v1/admins`
POST	`/admin/v1/admins`
PUT	`/admin/v1/admins/{id}/dis`
GET	`/admin/v1/system/operatio`

README.md

AI 平台接口文档

本文档覆盖平台所有业务模块的 REST API 接口定义，包含请求参数、响应结构、数据模型和使用示例。

文档导航

基础规范

文件	说明
00-overview.md	通用规范：Base URL、认证方式、统一响应格式、错误码体系、分页规范
00b-portal-guide.md	双端说明：用户控制台 vs 平台管理端，功能页面与 API 映射

文件	说明
00c-tech-stack.md	技术栈方案（已选定方案 C：Python FastAPI 全栈）
00d-preparation-checklist.md	上线前准备清单（域名/备案/第三方凭证/基础设施）
00e-baofoo-sms-api.md	宝付短信渠道接口（三环境地址 + 参数说明 + Python 示例）

用户与认证

文件	模块	主要功能
01-user-api.md	用户模块	注册、个人信息、实名认证、企业认证、企业成员管理、API Key

文件	模块	主要功能
02-auth-api.md	授权模块	登录/登出、Token 刷新、第三方 OAuth、账号绑定、企业 SSO、RBAC

财务

文件	模块	主要功能
03-wallet-api.md	钱包模块	余额查询、（支付宝/微信/宝付代金券、发票申请
04-transaction-api.md	交易模块	交易记录、交易详情、统计、导出
05-order-api.md	订单模块	创建订单、支付、取消退款、状态
06-billing-api.md	计费模块	计费规则、用量统计、

文件	模块	主要功能
		月度账单、费用预估
07-accounting-api.md	账务模块	收支报表、总账流水、对账管理、增值税发票

会员与营销

文件	模块	主要功能
08-member-api.md	会员模块	套餐列表、订阅、续费取消、会员
14-marketing-api.md	营销模块	代金券活动券类型、批量生成、自动发放规手动发放、

模型与 Token

文件	模块	主要功能
10-token-api.md	Token 模块	用量仪表盘、API Key 统计、Top 10、平台定价、第三方比价
11-model-api.md	模型模块	提供商管理、模型管理、对话（SSE 流式）、限流、请求队列、降级策略

平台基础设施

文件	模块	主要功能
09-gateway-api.md	API 网关	路由规则、全局限流、访问日志、Webhook、审计日志

文件	模块	主要功能
12-notification-api.md	通知模块	通知记录、偏好设置、广播、模板管理
13-message-center-api.md	消息中心	收件箱、分类、搜索、SSE 实时推送、公告管理

快速开始

认证方式

所有接口（除登录/注册外）均需携带认证头：

```
# 用户界面操作（JWT Token）
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

# 程序调用模型接口（API Key）
```

```
Authorization: Bearer sk-prod-x
xxxxxxxxxxxxxx
```

基础 URL

端	Base URL
用户控制台 API	`https://api.example`
平台管理端 API	`https://api.example`

统一响应格式

```
{
  "code": 0,
  "message": "success",
  "data": {},
  "request_id": "req_550e8400-e
29b-41d4-a716",
  "timestamp": 1704067200
}
```

错误码速查

范围	分类
`1xxx`	认证 / 授权错误
`2xxx`	用户业务错误
`3xxx`	支付 / 钱包错误
`4xxx`	模型 / Token 错误
`5xxx`	网关 / 限流错误
`6xxx`	账务 / 计费错误

详见 00-overview.md。

模块关系图

```
用户模块 ──────── 授权模块（登录/OAuth/RBAC）
    │
    ├──── 钱包模块 ──────── 交易模块 ──────── 账务模块
    │                          └──────────── 订单模块 ──────── 计费模块
    │
    ├──── 会员模块 ──────── 营销模块（代金券）
    │
```

```
      ├──── Token 模块 ──────── 计费模块
      │
      ├──── 模型模块（提供商 → 模型 → 对话 → 限流/队列/降级）
      │
      ├──── API 网关（路由 / 限流 / Webhook / 审计）
      │
      ├──── 通知模块（邮件 / 短信 / 站内信）
      │
      └──── 消息中心（收件箱 / 公告）
```

变更记录

版本	日期	说明
v1.0	2024-01	初始版本，覆盖 14 个业务模块

V2

规则

Python项目规则

.cursor/rules

你正在参与一个 Python 项目，必须严格遵守以下目录结构和开发规范：

- /src
    - /myproject # 主 Python 包（请将 “myproject” 替换为你实际的包名）
        - /api # API 路由、视图、请求/响应模型（如 FastAPI 路由器）
        - /core # 核心配置，如日志、安全、依赖注入、应用初始化
        - /db # 数据库相关：模型（models）、CRUD 操作、会话管理
        - /schemas # Pydantic 模型，用于数据校验与序列化（请求体、响应体）
        - /services # 业务逻辑层（例如用户注册、订单处理等核心逻辑）
        - /utils # 通用工具函数（如日期处理、字符串工具、文件操作等）

        - /tasks # 后台任务（如 Celery 任务）
        - /exceptions # 自定义异常类
        - init.py # 使目录成为 Python 包
        - main.py # 应用入口（例如创建 FastAPI 实例）
- /tests
    - /unit # 单元测试（使用 mock 隔离依赖）
    - /integration # 集成测试（连接真实数据库或外部服务）
    - conftest.py # pytest 全局 fixture 配置
- /scripts # 辅助脚本（如数据迁移、环境初始化、批量处理）
- /alembic # 数据库迁移脚本（如果使用 Alembic）
- /docs # 项目文档（Markdown 或 Sphinx 生成）
- /data # 示例数据或静态资源（禁止存放敏感信息）
- .env.example # 环境变量模板（真实 .env 文件不得提交到 Git）
- .gitignore
- pyproject.toml # 推荐的项目配置文件（替代 setup.py）
- requirements.txt # 依赖列表（若使用 pip）；若用 Poetry/Pipenv 则对应 lock 文件
- README.md
- Dockerfile # 若项目容器化
- docker-compose.yml # 若使用多服务编排

【强制规则】

1. 所有应用代码必须放在 `/src/myproject` 下，禁止在项目根目录写业务逻辑。
2. 业务逻辑必须写在 `/src/myproject/services`，不能直接写在 API 路由或数据库模型中。
3. 数据库模型放在 `/src/myproject/db/models.py`，CRUD 操作放在 `crud.py`。
4. API 路由应保持“薄”——只做参数校验和调用 service，不包含复杂逻辑。
5. 测试目录结构应与源码结构一致。
   例如：`src/myproject/services/user_service.py` → `tests/unit/services/test_user_service.py`
6. 禁止硬编码密钥或敏感信息，必须通过环境变量加载（推荐使用 Pydantic Settings）。
7. 包内使用相对导入（如 `from . import utils`），不要修改 `sys.path`。
8. 所有新模块目录必须包含 `__init__.py` 文件，确保可被导入。
9. 除非用户明确要求，否则不要生成 Jupyter Notebook（.ipynb）文件。
10. 如果不确定文件该放哪里，请优先考虑 `/src/myproject/utils`，或主动询问用户。

在生成代码、重构、补全或提供建议时，必须始终遵循以上结构和规则。

Ai-hub数据库表结构

数据库表

ai-hub

公共层（common）

存放一些工具类（IP、URL、系统日志、JSON、分页、邮件等….）

常量（constant）

该目录仅用于放置全局可复用的常量定义，不包含任何业务逻辑或依赖关系。

当前文件

文件	说明
`api_type.go`	定义与模型
`azure.go`	定义与 Azur `AzureNoRe` 的截止时间
`cache_key.go`	缓存键格式 相关字段常

文件	说明
`channel_setting.go`	Channel 级 `proxy`、
`context_key.go`	定义 `Conte` 类型以及在 （请求时间 相关信息等
`env.go`	环境配置相 在启动阶段
`finish_reason.go`	OpenAI/GP 字符串常量
`midjourney.go`	Midjourney 常量与模型
`setup.go`	标识项目是 布尔值)。
`task.go`	各种任务(Ta 动作常量及 Midjourney
`user_setting.go`	用户设置相 (Email/Web
…	…

接入层（controller）

计费

渠道

渠道计费

渠道策略（缓存和更新）

Token (API-KEY)

组

分组倍率配置：进入平台「系统管理 - 分组与模型定价设置」页签，在「分组倍率」栏中，以 JSON 字符串格式新增分组或修改现有分组的计费倍率，示例：{“vip”: 0.5, “test”: 1}，分别对应vip分组 0.5 倍、test分组 1 倍。

- 默认分组规则：新用户注册时，系统自动将其归属default分组。
- 分组修改权限：仅管理员及超级管理员拥有修改用户所属分组的权限。
- 默认选择分组

用户在创建 Token / 密钥时，下拉框里默认显示、可以直接选择的分组列表。

用户

定价（Pricing）【重点】

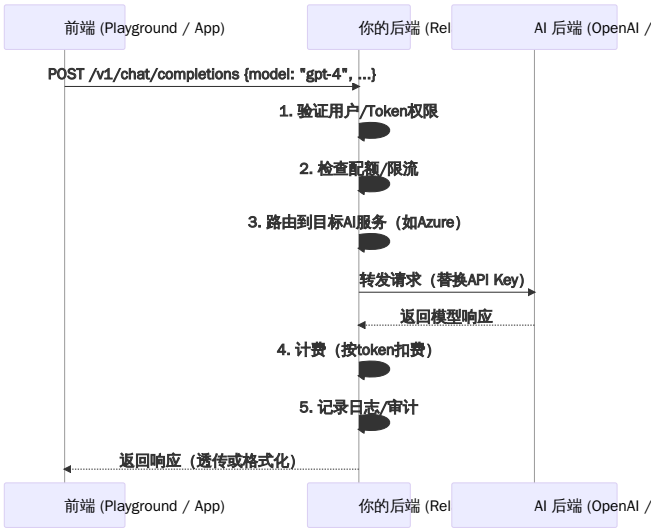
价格倍率（ratio-config、ratio_sync）

https://models.dev/

兑换（Redemptions）

中继

?????? 调用模型代码。



任务

第三方账号绑定（Telegram）

- 微信
- github
- google
- ….

订阅（SubscriptionPlanD）

设置token套餐
– 支付

预填组（PrefillGroups）
[待确认]

安全验证

- 2FA
- Passkey

系统参数

性能统计 （Performance）

统计当前项目的 CPU、内存、磁盘

初始化安装

签到

文档（DOCS）

数据传输类（DTO）

国际化（服务端）

中间层

通用、可复用的请求处理逻辑，用于在 HTTP 请求到达最终处理函数（Handler）之前或之后执行一些公共操作

model（数据库）

数据库操作对象

第三方授权API

- discord
- generic
- github
- linuxdo
- ….

pkg

放一些工具类

- 缓存
- IO

中继业务 （relay）

实际处理业务

路由（router）

- API
- 平台
- 中继
- 视频

- Web

业务类

音频

计费

Token 通用计算（文本 + 多媒体）

- 文本 Token

  纯字符模式：直接取字符数

  其他：调用模型专用算法计算

- OpenAI 格式额外加成

  工具数 ×8 + 消息数 ×3 + 名称数 ×3 + 固定 3 Token

- 多媒体 Token（固定值）

  图片：OpenAI 模型精算 / 其他模型 + 520

  音频：+256

视频：+8192

文件 / 未知类型：+4096

费用计算

1. 获取当前模型的价格（modelPrice）和是否启用价格模式（usePrice）；
2. 处理请求的群组倍率信息（groupRatioInfo），用于后续配额计算。

分模式计算预消耗配额（核心逻辑）

- 模式1：不启用价格模式

  （usePrice = false）

1. 计算预消耗 Token 数：取提示词 Token 数（promptTokens）与默认预消耗配额的最大值，若存在最大 Token 数（meta.MaxTokens）则叠加；
2. 获取模型倍率（modelRatio），若未配置且未开启“允许未配置倍率”，则返回错误；
3. 依次获取完成倍率、缓存倍率、缓存创建倍率（5分钟与1小时倍率

按固定比例计算）、图片倍率、音频相关倍率；

4. 计算预消耗配额：预消耗 Token 数 × 模型倍率 × 群组倍率。

- 模式2：启用价格模式（usePrice = true）

1. 若存在图片价格倍率（meta.ImagePriceRatio），则用该倍率修正模型价格；
2. 计算预消耗配额：模型价格 × 单位配额系数 × 群组倍率。

三、免费模型配额管控

若全局关闭免费模型预消耗功能，满足以下任一条件则预消耗配额设为0，标记为免费模型（freeModel = true）：

1. 群组倍率为0；
2. 价格模式下，模型价格为0；
3. 非价格模式下，模型倍率为0。

计费Session

单次请求的预扣费/结算/退款生命周期

渠道

渠道策略

渠道选择

Codex授权

Codex证书刷新

Codex证书刷新任务

codex负载使用量

第三方支付（epay、…）

文件处理

资金来源

组

图片

模型日志

限额通知

openIA接口

限额

敏感词

订阅

任务

任务计费

任务适配

文本限额

Token

- 不同厂商的计费权重
- TOKEN版面
- 初始化默认模型

使用量

用户通知

违规费用

Webhook

设置

缓存这系统的参数

模型设置

- claude
- gemini
- global
- grok
- qwen

操作设置

- 渠道策略
- 支付
- 限额
- token
- 工具

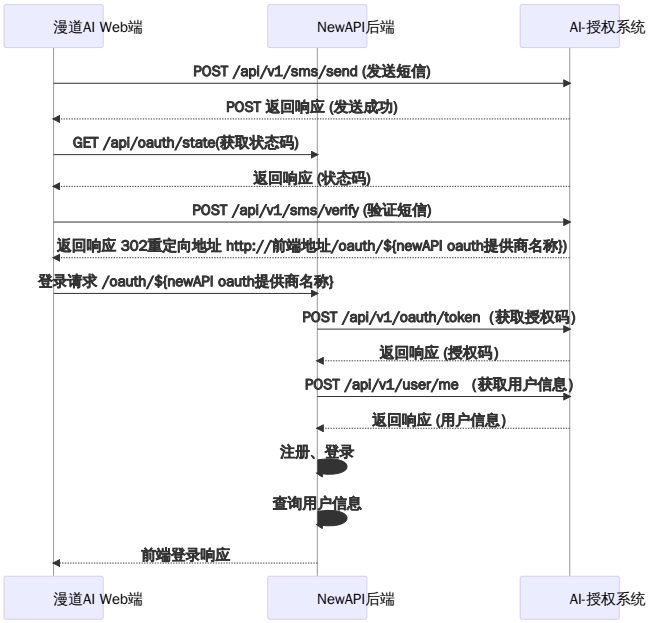
枚举

前端

需要补充需求

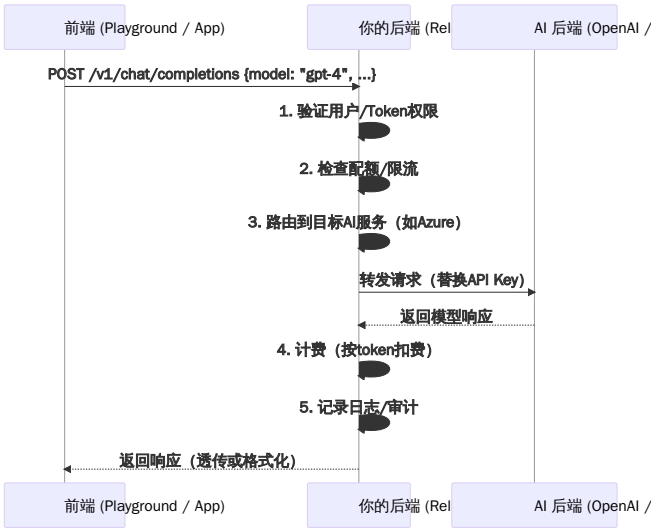
短信

短信登录


漫道AI Web端
NewAPI后端
AI-授权系统
POST /api/v1/sms/send (发送短信)
POST 返回响应 (发送成功)
GET /api/oauth/state(获取状态码)
返回响应 (状态码)
POST /api/v1/sms/verify (验证短信)
返回响应 302重定向地址 http://前端地址/oauth/${newAPI oauth提供商名称})
登录请求 /oauth/${newAPI oauth提供商名称}
POST /api/v1/oauth/token（获取授权码）
返回响应 (授权码)
POST /api/v1/user/me （获取用户信息）
返回响应 (用户信息)
注册、登录
查询用户信息
前端登录响应
漫道AI Web端
NewAPI后端
AI-授权系统

注册

重要业务流程


前端 (Playground / App)
你的后端 (Rel
AI 后端 (OpenAI /
POST /v1/chat/completions {model: "gpt-4", ...}
1. 验证用户/Token权限
2. 检查配额/限流
3. 路由到目标AI服务（如Azure）
转发请求（替换API Key）
返回模型响应
4. 计费（按token扣费）
5. 记录日志/审计
返回响应（透传或格式化）
前端 (Playground / App)
你的后端 (Rel
AI 后端 (OpenAI /

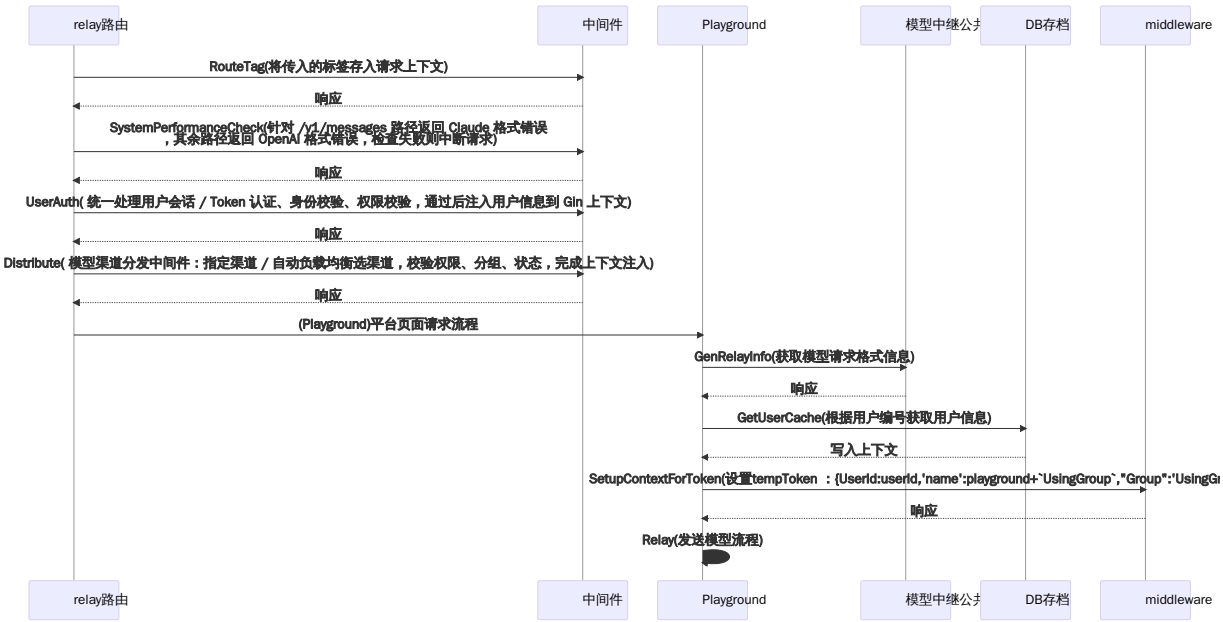
对比

维度	NewAPI	企业级
权限与多租户	令牌	租户
计费与成本	按调用次数 / Token、缓存计费、预付费	多维度（部门预算熔资源优
安全与合规	基础鉴权、审计日志	基础鉴

ai-hub-chart

对话流程

页面



relayRouter->>Middleware: Distribute

核心功能拆解

1. 两种渠道分配方式

- 强制指定渠道：从上下文取固定渠道 ID，直接校验状态

- 自动分配渠道：根据请求模型、用户分组、偏好亲和性，随机选取健康可用渠道

2. 多层权限校验

- Token 模型权限：校验用户是否允许使用当前模型
- 分组权限：校验用户是否有权访问请求的分组
- 渠道状态：仅使用启用（Enabled）的渠道

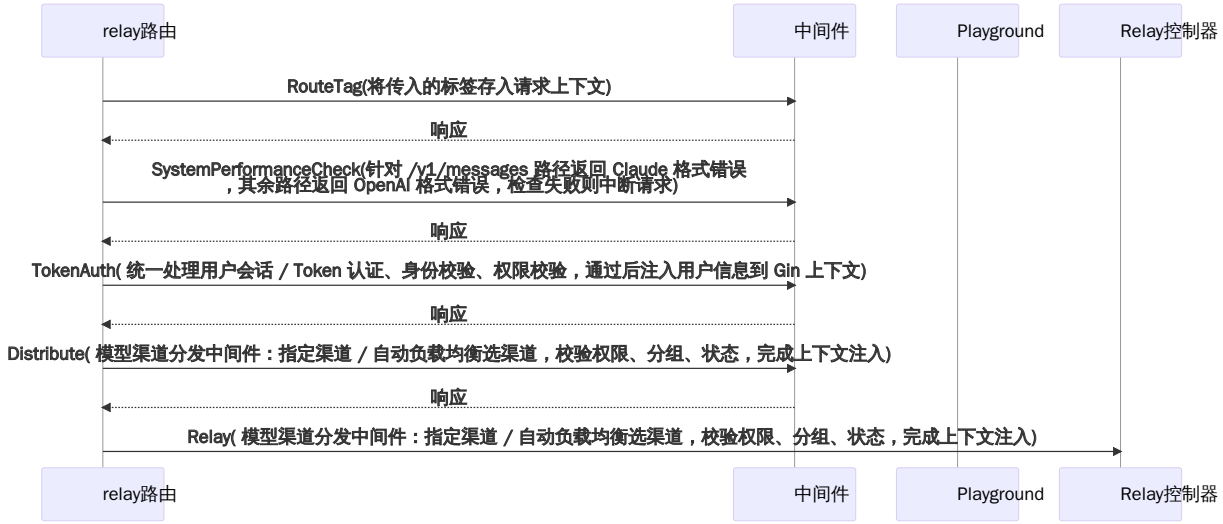
3. 特殊场景处理

- 支持 Playground 接口自定义分组
- 支持渠道亲和性（优先使用上次成功的渠道）
- 支持 auto 自动分组智能匹配

4. 收尾动作

- 记录请求开始时间
- 注入渠道信息到上下文
- 请求成功后记录渠道亲和性

通过API请求的类



relayV1Router->>TokenAuth: UserAuth

1. 兼容各种密钥位置

- WebSocket 从协议头取 key
- Claude 从 x-api-key 取
- Gemini 从 URL 参数 / 请求头取

2. 统一转成标准 Bearer 密钥

- 校验令牌合法性
- 解析密钥 → 验证有效性
- 校验用户是否被封禁
- 校验 IP 是否在白名单

3. 校验权限并放行

- 校验用户分组、令牌分组权限

- 把用户 / 令牌信息存入上下文
- 放行后续业务逻辑

API

文本视频生成

阿里云

模型价格

模型	价格	备注
Wan2.6-T2V	0.6元/秒 720p、1元/秒 1080p	-
Wan2.5-T2V-Preview	480p 0.3元/秒、 720p 0.6元/秒、1元/秒 1080p	-
PixVerse-V6-t2v	60P 0.21元/每秒 540P 0.27元/每秒 720P 0.36元/每秒 1080P 0.68元/每秒 360P 无声 0.15元/每秒 540P 无声 0.21元/每秒 720P 无声 0.27元/每秒 1080P 无声 0.53元/每秒	-
PixVerse-V5.6-t2v	360P 0.47元/每秒 540P 0.47元/每秒 720P 0.53元/每秒 1080P 0.7元/每秒 360P 无声 0.21元/每秒 540P 无声 0.21元/每秒	-

模型	价格	备注
	720P 无声 0.27元/每秒 1080P 无声 0.44元/每秒	
Kling Video 3.0	720P 0.9元/每秒 1080P 1.2元/每秒 720P 无声 0.6元/每秒 1080P 无声 0.8元/每秒	
Kling Video 3.0 Omni	720P 0.9元/每秒 1080P 1.2元/每秒 720P 无声 0.6元/每秒 1080P 无声 0.8元/每秒 720P 无声 0.9元/每秒 1080P 无声 1.2元/每秒	-
ViduQ3-Pro_text2video	540P 0.3125元/每秒 720P 0.78125元/每秒 1080P 0.9375元/每秒	
ViduQ3-Turbo_text2video	540P 0.25元/每秒 720P 0.375元/每秒 1080P 0.4375元/每秒	
ViduQ2_text2video	540P 0.1125元/每秒 720P 0.21875元/每秒 1080P 0.375元/每秒	
Wan2.2-T2V-Plus	480P 0.14元/每秒 1080P 0.7元/每秒	

模型	价格	备注
Wan2.1-T2V-Plus	720P 0.7元/每秒	生成时间：1分半
Wan2.1-T2V-Turbo	480P 0.24元/每秒 720P 0.24元/每秒	

模型区别

模型名称	所属公司/项目	发布时间	视频时长	特色
万相 2.6 / 2.5	阿里云（通义实验室）	2024年中（2.6为最新）	≤16s（2.5），≤22s（2.6）	多镜分镜高画支持
PixVerse V6 / V5.6	PixVerse（海外团队，中国有合作）	V6：2024年3月；V5.6：2023年底	≤4s（V5.6），≤6s（V6）	强调支持（推可加
Kling Video 3.0 / Omni	字节跳动（剪映团队）	2024年Q2（3.0），2024年Q3（Omni）	≤8s（3.0），≤16s（Omni）	“全能支持+文+视8秒自动

模型名称	所属公司/项目	发布时间	视频时长	特色
Vidu Q3-Pro / Turbo / Q2	生数科技（清华系，被百度投资）	Q2:2023底；Q3:2024Q2；Turbo:2024Q3	Q2:≤4s；Q3/Q3-Pro:≤8s；Turbo:≤16s	文生 +视 “精 “高 支持

腾讯

模型价格

模型	价格	备注
HY-Video-1.5	1.5积分	
YT-Video-2.0	2积分	
YT-Video-HumanActor	720p 1积分 1080p 2积分	
YT-Video-FX	720P 1.5积分 360p 1积分	

积分说明

1. 混元生视频（HY-Video-1.5）—— 你最可能的场景
   计费标准：1.5 积分 / 次腾讯云

官方汇率：后付费：1 积分 = 1.2 元腾讯云
计算：
1.5 积分 × 1.2 元 / 积分 = 1.8 元 / 次

2. 混元生 3D（HY-3D）—— 另一种积分计价
官方汇率：1 积分 = 0.12 元腾讯云
如果是 3D 场景误看：
1.5 积分 × 0.12 元 / 积分 = 0.18 元 / 次

模型区别

模型名称	全称/类型	核心定位
HY-Video-1.5	混元生视频（基础版）	通用文生视频（Text-to-Video）主干模型 支持输入中文提示词生成高质量短视（默认约4–8秒）
YT-Video-2.0	园生视频（推测为“元视频”或“园”为内部代号）	进阶版文生视频模型 相比1.5版：更长时长（可达12–16s）、更高分辨率、 更强运动连贯性、支持复杂场景（如多人互动、动态背景）
YT-Video-HumanActor	人像驱动（Avatar/数字人驱动）	专为人像/数字人视频生成设计 输入文本 + 参考图/语音 → 生成指定人物口播、表情、 动作一致的视频 适用于虚拟主播、AI讲师、客服等

模型名称	全称/类型	核心定位
YT-Video-FX	视频特效增强	视频后处理/特效生成模型 非直接文生视频， 而是对已有视频添加特效： • 动态滤镜 / 魔法转场 / 粒子效果 / 3D风格化 • 可配合其他模型使用 （先用HY生成视频，再用FX加特效）

抖音

模型价格

模型	音画同步	成本	推荐
Seedance-2.0	完美同步	0.046 元 / 千 tokens	-
Seedance-2.0-fast	同步	0.037 元 / 千 tokens	-
Seedance-1.5-pro	同步	0.016 元 / 千 tokens	-
Seedance-1.0-pro	无声	15元/百万token 5秒 480p	-

模型	音画同步	成本	推荐
		预计消耗 48,600 Tokens (864宽*480高*24帧*5s)/1024*1条 0.729 元 720p 预计消耗 102,960 Tokens (1248宽*704高*24帧*5s)/1024*1条 1.5444 元 1080p 预计消耗 244,800 Tokens (1920宽*1088高*24帧*5s)/1024*1条 3.672 元	
Seedance-1.0-pro-fast	无声 10 秒	同上	-

华为

- 使用阿里的万相模型

百度

百度蒸汽机 Air

智谱

CogVideoX-3

模型	部署版本	私有实例
Cogvideox	文生视频-int8	100 元 / 算力单元 / 天
Cogvideox	图生视频-int8	100 元 / 算力单元 / 天

月之暗面

Kimi-VL

使用场景

使用“中景，两个孩子在花园里追逐蝴蝶，明亮自然光”生成视频

阿里云

模型

Wan2.6-T2V

- 价格 ：0.6元/秒 720p 1元/秒 1080p
- 生成时间：生成时间1分-1.5分钟

Wan2.1-T2V-Plus（推荐）

- 价格 ：0.7元/秒 720p
- 生成时间：生成时间1分-1.5分钟

访问地址

https://bailian.console.aliyun.com/cn-beijing?spm=5176.smartservice_service_robot_chat_new.console-base_product-drawer-right.dproducts-and-services-sfm.15cef6254eIL0Z&tab=api#/api/?type=model&url=2862677

UCLOU 使用

使用- `阿里云` 模型Wan, 无法体验。

访问地址：

https://astraflow.ucloud.cn/modelverse/playground#all

腾讯云

HY-Video-1.5

- 价格: 1.5 积分
- 生成时间：生成时间3分

YT-Video-2.0

- 价格: 2.0积分
- 生成时间：生成时间2分

未知：

积分如何兑换金额

访问地址

https://console.cloud.tencent.com/tokenhub/creation

火山引擎

模型

Doubao-Seedance-2.0

- 价格 ：0.046元/千token 480p
- 生成时间：在豆包对话平台中：4-5分钟，在火山引擎平台中，10分钟，未成功。
- 备注：目前公测当中。

访问页面:

https://console.volcengine.com/ark/region:ark+cn-beijing/experience/vision?modelId=doubao-seedance-2-0-

260128&tab=GenVideo

AI平台与BIII对账

模型价格计算：

- 使用同一个模型 claude-opus-4-6
- 使用不同的提供商类型：

  Anthropic、OpenAI
- 与引用平台计算不同的价格

AI平台与BIII平台对账(使用Anthropic模型提供商，使用模型 claude-opus-4-6)

类型	AI平台	bill平台
总金额	51.53	51.063
总token	7115590	70501
输入token	86856	21427
输出token	61224	61205

类型	AI平台	bill平台
缓存写入token	755674	75567
缓存读取token	6211836	62118
请求次数	139	138

AI：
模型	输入	输出	缓存写入	缓存读取
claude-haiku-4-5-20251001	84130	23897	199898	1855075
claude-opus-4-6	2726	37327	555776	4356761

BILL平台：
模型	输入	输出	缓存写入	缓存读取
claude-haiku-4-5-20251001	18702	23897	199898	1855075
claude-opus-4-6	2725	37308	555776	4356761

claude-haiku-4-5-20251001 输入少记两笔：token数 (33269+32159) 计算差额：(33269+32159) / 1M tokens * ¥7.120000 = 0.465847
claude-opus-4-6：输入：1 输出:19

计算差额：`（33269+32159）/ 1M tokens * ¥7.120000 = 0.465847`
（NewAPI平台金额）51.53 - 51.0637 （bill平台金额） = 0.4663

记录详情

bill平台
2026/4/9 11:13:01 claude-haiku-4-5-20251001_output 输出 `1.095` ¥35.60 ¥0.0390 已扣费
2026/4/9 11:13:01 claude-haiku-4-5-20251001_cache_write 缓存写入 `0.921` ¥8.90 ¥0.0082 已扣费
2026/4/9 11:13:01 claude-haiku-4-5-20251001_cache_read 缓存读取 `42.701` ¥0.71 ¥0.0304 已扣费

2026/4/9 11:12:49 claude-haiku-4-5-20251001_output 输出 0.100 ¥35.60 ¥0.0036 已扣费
2026/4/9 11:12:49 claude-haiku-4-5-20251001_cache_write 缓存写入 0.756 ¥8.90 ¥0.0067 已扣费
2026/4/9 11:12:49 claude-haiku-4-5-20251001_cache_read 缓存读取 41.945 ¥0.71 ¥0.0299 已扣费

AI平台

1. 第一笔
   2026-04-09 11:13:01 claude-haiku-4-5-20251001 输入:33269 缓存读 42,701 · 写 921 输出:1095 ¥0.31
   提示 `33269` tokens / 1M tokens * ¥7.120000 + 缓存 `42701` tokens / 1M tokens * ¥0.712000 + 5m缓存创建 `921` tokens / 1M tokens * ¥8.900000 + 补全 `1095` tokens / 1M tokens * ¥35.600000 * 分组倍率 1 = ¥0.314457

2. 第二笔
   2026-04-09 11:12:49 claude-haiku-4-5-20251001 输入：32159 缓存读 41,945 写 756 输出:100 ¥0.27
   提示 `32159` tokens / 1M tokens * ¥7.120000 + 缓存 41945 tokens / 1M tokens * ¥0.712000 + 5m缓存创建 756 tokens / 1M tokens * ¥8.900000 + 补全 100 tokens / 1M tokens * ¥35.600000 * 分组倍率 1 = ¥0.269125

3. 第三笔
   2026-04-09 10:54:15 claude-opus-4-6 写：1 输出19 ¥0.01 输入价格 ¥35.600000 / 1M tokens，输出价格 ¥178.000000 / 1M tokens，缓存读取价格 ¥3.560000 / 1M tokens，缓存创建价格 ¥44.500000 / 1M tokens，分组倍率 1.0000x

AI平台与BIII平台对账(使用OpenAI模型提供商，使用模型 claude-opus-4-6)

时间段：2026/04/09 15:00 - 2026/04/09 18:30

类型	AI平台	bill平台
总金额	27.18	28.14
总token	1325408	12007
输入token	1057459	93277
输出token	17302	17,302
缓存写入token	44946	44946
缓存读取token	205701	20570

类型	AI平台	bill平台
请求次数	52	42

AI 平台计价 10笔 总： 124684，输出为空

AI平台与BIII平台对账(使用OpenAI模型提供商，使用模型 gpt-5.4)

使用时间：19:00 - 20:00

类型	AI平台	bill平台
总金额	9.73	9.3343
总token	527754	505747
输入token	523924	502017
输出token	3830	3730
缓存写入token	0	0
缓存读取token	0	0
请求次数	8	7

2026-04-09 19:11:06 OpenAI测试
gpt-5.4 21907 100 ¥0.40
(输入 21907 tokens / 1M tokens * ¥17.800000 + 输出 100 tokens / 1M tokens * ¥106.800000) * 分组倍率 1 = ¥0.400625

其他使用命令

```
SELECT
    model_name,
    sum( prompt_tokens ),
    sum( completion_tokens ),
    sum( cache_write_tokens ),
    sum( cache_tokens )
FROM
    (
    SELECT
        FROM_UNIXTIME(created_at) AS `created_at`,
        model_name,
        prompt_tokens AS `prompt_tokens`,
        completion_tokens AS `completion_tokens`,
        JSON_EXTRACT( other, '$.cache_write_tokens' ) AS `cache_write_tokens`,
```

```
        JSON_EXTRACT( other, '$.cache_tokens' ) AS `cache_tokens`
    FROM
        `logs`
    WHERE
        `group` = '渠道类型测试-吴升斌OpenAI'
        AND content = ''
        AND created_at >= UNIX_TIMESTAMP(
        CURDATE())
        AND created_at < UNIX_TIMESTAMP(
        DATE_ADD( CURDATE(), INTERVAL 1 DAY ))
    ) t
GROUP BY
    model_name;


------

SELECT

    model_name,
    sum(prompt_tokens) AS `prompt_tokens`,
    sum(completion_tokens) AS `
```

```
completion_tokens`,
    sum(JSON_EXTRACT(other, '$.
cache_write_tokens')) AS `cache
_write_tokens`,
    sum(JSON_EXTRACT(other, '$.
cache_tokens')) AS `cache_token
s`
FROM
    `logs`
WHERE
    `model_name` = 'qwen3.7-ma
x'
    AND created_at >= UNIX_TIME
STAMP(DATE_SUB(CURDATE(), INTER
VAL 1 DAY))
    AND created_at < UNIX_TIMES
TAMP(CURDATE())
group BY
   model_name

---

SELECT
FROM_UNIXTIME( created_at ) AS
`created_at`,
model_name,
prompt_tokens AS `prompt_tokens
`,
completion_tokens AS `completio
```

```
n_tokens`,
JSON_EXTRACT( other, '$.cache_write_tokens' ) AS `cache_write_tokens`,
JSON_EXTRACT( other, '$.cache_tokens' ) AS `cache_tokens`
FROM
    `logs`
WHERE
    `group` = '渠道类型测试-吴升斌OpenAI'
    AND content = ''
    AND created_at >= UNIX_TIMESTAMP(
    CURDATE())
    AND created_at < UNIX_TIMESTAMP(
    DATE_ADD( CURDATE(), INTERVAL 1 DAY ))
    ORDER BY created_at desc



SELECT
    sum(prompt_tokens) AS `prompt_tokens`,
    sum(completion_tokens) AS `completion_tokens`,
    sum(JSON_EXTRACT( other,
```

```
'$.cache_write_tokens' )) AS `cache_write_tokens`,
    sum(JSON_EXTRACT( other, '$.cache_tokens' )) AS `cache_tokens` ,
    sum(JSON_EXTRACT( other, '$.image_output' )) AS `image_output`
FROM
    `logs`
WHERE
      type = 2
    AND created_at >= 1783468800
    AND created_at < 1783555200
```

openAI-Anthropic

从 controller/relay.go 的 Relay 入口往下串了一遍，核心链路是：

- controller/relay.go:67 Relay
- dto/openai_request.go:119 GeneralOpenAIRequest.GetTokenCountMeta
- dto/claude.go:234 ClaudeRequest.GetTokenCountMeta
- service/token_counter.go:183 EstimateRequestToken
- service/token_counter.go:399 CountTextToken
- service/token_estimator.go:68 EstimateToken
- relay/helper/price.go:48 ModelPriceHelper
- 返回阶段：
  - OpenAI: relay/channel/openai/relay-openai.go:106, relay/channel/openai/relay-openai.go:195
  - Anthropic: relay/channel/claude/relay-claude.go:24, relay/claude_handler.go:815, relay/claude_handler.go:882

  下面按“请求预估”和“响应实际 usage”两部分总结 OpenAI 和 Anthropic 的区别。

1. 请求进入时，两者怎么估算 prompt token

   共性

   在 controller/relay.go:125-152：

2. 先从 request 提取 TokenCountMeta

3. 调 service.EstimateRequestToken(…)

4. 把结果存到 relayInfo.SetEstimatePromptTokens(tokens)

5. 再交给 relay/helper/price.go:48 用于预扣费

   也就是说，OpenAI 和 Anthropic 在入口层都先做一次“本地预估”，用于风控/预扣费。

---

2. OpenAI 的 token 计算方式

   2.1 文本部分：尽量用真正 tokenizer

   在 service/token_counter.go:399：

- 如果是 OpenAI 文本模型，common.IsOpenAITextModel(model) 为真
- 就走 getTokenEncoder(model) + getTokenNum(…)
- 也就是真正按 tokenizer 算

  代码位置：

- service/token_counter.go:404-406

  这点是 OpenAI 和 Anthropic 最大区别之一。

  2.2 额外加上 OpenAI chat message 格式开销

  在 service/token_counter.go:230-235：

  如果 info.RelayFormat == types.RelayFormatOpenAI，还会额外加：

- meta.ToolsCount * 8
- meta.MessagesCount * 3
- meta.NameCount * 3
- 最后再 +3

  这明显是在模拟 OpenAI chat/completions 的消息包装成本。

  代码位置：

- service/token_counter.go:230-235

  2.3 OpenAI 请求元数据提取更偏向 Chat 格式

  在 dto/openai_request.go:119-224，GetTokenCountMeta() 会提取：

- messages 中的 role
- name
- 文本内容
- tools 的 name/description/parameters
- 图片/音频/文件/视频等多模态内容
- MaxTokens / MaxCompletionTokens

  并且会统计：

- MessagesCount
- NameCount
- ToolsCount

  这些字段后面会直接参与 OpenAI 的额外 token 开销计算。

  2.4 多模态图片对 OpenAI 是特殊精算

  在 service/token_counter.go:22 的 getImageToken(…)：

- 对 OpenAI 系列模型（如 4o/4.1/o1/o3/gpt-5 等）按模型规则精算图片 token
- 包括 patch-based / tile-based 两套算法

- 不是简单固定值

  而在主流程里：

- service/token_counter.go:279-285

- 只有 common.IsOpenAITextModel(model) 才走 getImageToken(…)

  也就是说，OpenAI 图片 token 估算明显更细。

---

3. Anthropic 的 token 计算方式

   3.1 文本部分：不走 tokenizer，走估算器

   在 service/token_counter.go:404-410：

- 非 OpenAI 模型不会走 tokenizer

- 直接走 EstimateTokenByModel(model, text)

  如果模型名包含 claude，则走：

- service/token_estimator.go:225-226

  最终使用的是：

- service/token_estimator.go:68 EstimateToken(Claude, text)

这是一套基于字符类别权重的估算算法，不是 Claude 官方 tokenizer。

3.2 Claude 有自己一套估算权重

在 service/token_estimator.go:40-42，Claude 的权重是：

- Word: 1.13
- Number: 1.63
- CJK: 1.21
- Symbol: 0.4
- MathSymbol: 4.52
- URLDelim: 1.26
- AtSign: 2.82
- Emoji: 2.6
- Newline: 0.89
- Space: 0.39

对比 OpenAI：

- OpenAI 的 CJK 更低：0.85

- Claude 的 CJK 更高：1.21
- Claude 的 Emoji / MathSymbol / @ 更高
- OpenAI 的换行更低：0.5

所以从代码设计看，项目认为 Claude 对中文、emoji、数学符号、@ 这类内容更“吃 token”。

3.3 Anthropic 请求没有加 OpenAI 那套 message 包装常数

service/token_counter.go:230-235 那段只对 RelayFormatOpenAI 生效。

所以 Claude 请求不会额外加：

- 每条消息 +3
- 每个 name +3
- 工具 *8
- 收尾 +3

也就是说，Anthropic 预估主要靠拼出来的 CombineText 本体，不加 OpenAI chat 协议的固定格式开销。

3.4 Claude 的 meta 提取更贴近 Anthropic 消息结构

在 dto/claude.go:234-360：

会提取：

- system
  - 如果是 string，直接取
  - 如果是多段 media，也会取 text/image
- messages
  - role
  - text
  - image
  - tool_use
  - tool_result
- tools
  - 普通 tool 的 name/description/input_schema
  - web search tool 的 name/user_location

特点是：

- Claude 把 system 也明确算进去了
- tool_use / tool_result 的内容也被拼进文本估算

- 结构更符合 Anthropic messages API

---

4. 多模态计算差异

   图片

   在 service/token_counter.go:276-297：

- OpenAI 模型图片：走 getImageToken(…) 精算
- 非 OpenAI 模型图片：直接 +520

   所以 Claude 图片 token 在请求预估阶段是：

- 固定近似值 520
- 不是 Claude 专属图片 tokenizer

   这点很重要：Claude 的图片 token 预估明显比 OpenAI 粗糙。

   音频 / 视频 / 文件

   统一是粗略固定值：

- audio: +256
- video: +4096 * 2
- file: +4096

OpenAI 和 Anthropic 这里差异不大，主要差在图片。

---

5. 响应返回后，实际 usage 的来源区别

   这部分比请求预估更关键，因为最终结算更依赖实际 usage。

5.1 OpenAI：优先信上游 usage，缺失时本地补算

OpenAI 处理在：

- 流式：relay/channel/openai/relay-openai.go:106
- 非流式：relay/channel/openai/relay-openai.go:195

  非流式

  在 relay/channel/openai/relay-openai.go:243-258：

  如果上游返回的 usage.prompt_tokens == 0，就兜底：

- prompt = info.GetEstimatePromptTokens()
- completion = 从响应文本再数一次
- total = 两者相加

  即：

  OpenAI 正常情况以 upstream usage 为准；没有 usage 才 fallback 本地估算。

流式

在 relay/channel/openai/relay-openai.go:183-186：

如果流里没带 usage，就：

- service.ResponseText2Usage(…)
- prompt 用 info.GetEstimatePromptTokens()
- completion 用输出文本再数

所以 OpenAI 的策略是：

- 优先使用上游 usage
- 缺了再用本地 tokenizer / 文本补算

5.2 Anthropic：usage 语义是 Anthropic 原生，再按需要映射成 OpenAI 风格

Anthropic 处理在：

- 流式：relay/claude_handler.go:815
- 非流式：relay/claude_handler.go:882

Claude 原生 usage 提取

在 relay/claude_handler.go:672-681 和 691-715：

Claude usage 字段来自上游：

- InputTokens
- CacheReadInputTokens
- CacheCreationInputTokens
- OutputTokens

并写入本地 dto.Usage：

- PromptTokens = InputTokens
- CompletionTokens = OutputTokens
- PromptTokensDetails.CachedTokens = CacheReadInputTokens
- PromptTokensDetails.CachedCreationTokens = CacheCreationInputTokens

同时标记：

- UsageSemantic = “anthropic”

流中如果没有完整 usage

在 relay/claude_handler.go:778-795：

如果流结束时：

- PromptTokens == 0，先用入口预估 estimatePromptTokens 兜底
- CompletionTokens == 0 或流未正常完成，再用 service.ResponseText2Usage(…) 按文本补算

  所以 Anthropic 也是：

- 优先用上游 usage
- 缺失时用本地估算兜底

  但 Anthropic 有缓存 token 的专门处理

  这是 OpenAI 处理里没有这么重视的一点。

  在：

- relay/claude_handler.go:558-585
- relay/claude_handler.go:572-584

  有专门逻辑把 Anthropic cache token 映射成 OpenAI 风格 usage：

- cache_read_input_tokens
- cache_creation_input_tokens
- claude_cache_creation_5_m_tokens

- claude_cache_creation_1_h_tokens

  并在映射时重新计算：

- PromptTokens

- InputTokens

- TotalTokens

  也就是：
  Anthropic usage 不只是 input/output token，还包含 cache creation / cache read 语义，项目里专门做了兼容转换。

---

6. 一句话概括两者差异

   OpenAI

- 请求阶段：尽量用真实 tokenizer

- 还会加上 OpenAI chat 消息格式固定开销

- 图片 token 按模型规则精算

- 响应阶段：优先信 upstream usage，缺失再本地补算

  Anthropic

- 请求阶段：不用 tokenizer，按 Claude 专属权重估算

- 不加 OpenAI chat 的固定包装 token
- 图片 token 多数情况下用固定近似值 520
- 响应阶段：优先信 Anthropic 原生 usage
- 且会额外处理 cache read / cache creation，再按需要映射成 OpenAI 风格 usage

---

7. 你可以重点记住的“本质区别”

   最核心就三条：

8. OpenAI 文本 token 是“真 tokenizer”计算；Anthropic 文本 token 是“权重估算”
    - service/token_counter.go:404-410
9. OpenAI 会加 chat 协议格式开销；Anthropic 不会
    - service/token_counter.go:230-235
10. Anthropic 的实际 usage 里更强调 cache token 语义
    - relay/claude_handler.go:572-585

如果你要，我也可以下一条继续给你整理成一张“对比表”，按：

- 请求体提取字段

- 预估算法
- 图片处理
- 上游 usage 兜底
- 缓存 token 处理

  做成更直观的表格版。

AI平台使用使用资金

cursor平台

美元：20 （3月） + 20 （超出使用） = 40
token: 235217873

sk-1GWWGVbJ2o0jgRMP0AUfbmdC-nwMmASsB7bQxLWzOIA
使用：400.4870元
token: 11703895

sk-yliykJ55KpZnVcQe83vDPRAL6jR8WlpK9vjybO1MRDg
使用：51.0637
token: 7050142

测试环境：
金额：609.79
token：19159475

总计：
token：235217873+19159475+7050142+11703895=273,131,385
金额：（20+20） * 7 + 400.4870+ 51.0637 +609.79=1,341.3407

吴升斌：
工具： 20美元 * 2 = 280 元
ucloud：50W token
token总数：2.7 亿

工作量-企业角色

增加企业级角色的影响分析

当前角色体系

角色是数值层级结构，值越大权限越高：

常量	值	说明
RoleGuestUser	0	访客
RoleCommonUser	1	普通用户
RoleAdminUser	10	管理员
RoleRootUser	100	超级管理员

系统大量使用 >= 比较来判断权限，比如 role >= 10 即为管理员。因此新角色的数值选择直接决定了它拥有哪些权限。
需要变动的地方（共 7 大块）

1. 常量定义 — common/constants.go

- 新增 RoleEnterpriseUser = ?（如 5 介于普通和管理员之间，或 50 介于管理员和 Root 之间）
- 更新 IsValidateRole() 函数加入新角色

2. 后端权限中间件 — middleware/auth.go

- authHelper() 用 role < minRole 做层级判断，数值本身会自动生效
- 但可能需要新增一个 EnterpriseAuth() 中间件工厂，类似 AdminAuth() / RootAuth()
- validUserInfo() 调用了 IsValidateRole()，改了常量就自动生效

3. 用户管理控制器 — controller/user.go

   这是改动最多的地方：

- 角色层级校验（约 6 处）：myRole <= user.Role 的比较，决定谁能管理谁
- promote/demote 操作（L900-919）：当前只支持 common ↔ admin，需新增企业角色的升降级逻辑
- 用户创建（L828）：user.Role >= myRole 的校验
- 权限计算 calculateUserPermissions()（L424-449）：需定义企业角色能看到哪些功能

- 侧边栏配置 generateDefaultSidebarConfig()（L452-511）：需定义企业角色的默认侧边栏

4. 用户模型 — model/user.go

- IsAdmin()（L716）：role >= RoleAdminUser — 企业角色是否算管理员？
- generateDefaultSidebarConfigForRole()（L99-158）：需增加企业角色的 if 分支
- GetRootUser() / RootUserExists() 不受影响

5. 其他后端引用

- service/user_notify.go:29：通知查询 role >= RoleAdminUser，企业角色是否需要收到管理通知？
- controller/group.go:38：role == RoleCommonUser 限制切组，企业角色要不要允许？
- controller/twofa.go:524：2FA 管理的角色权限校验

6. 前端角色判断 — 硬编码数值

- web/src/helpers/utils.jsx — isAdmin() 用 role >= 10，isRoot() 用 role >= 100
- web/src/helpers/auth.jsx:59 — AdminRoute 用 role >= 10

- web/src/components/layout/SiderBar.jsx — 多处调用 isAdmin() / isRoot() 控制菜单显隐
- web/src/components/settings/personal/components/UserInfoHeader.jsx — 显示角色标签
- web/src/pages/Setting/index.jsx:60 — 系统设置页只对 isRoot() 可见
- 需要新增 isEnterprise() 工具函数，或调整现有判断逻辑

7. 路由层 — router/api-router.go

- 管理路由用 AdminAuth() 保护，可能需要新增企业角色专属路由或调整中间件

关键决策点（编码前必须确定）

1. 数值选多少？ — 决定了它在层级中的位置
    - 5：在普通用户之上、管理员之下（有部分管理能力但不是完整管理员）
    - 50：在管理员之上、Root 之下（比管理员权限更大）
2. 企业角色能做什么？ — 比如：
    - 能否看到/管理渠道、模型、兑换码、用户？
    - 能否访问系统设置？
    - 能否切换分组？
    - 是否接收管理通知？
3. 谁能分配企业角色？ — 只有 Root？还是 Admin 也可以？

4. promote/demote 流程怎么走？ — common → enterprise → admin → root 四级，还是只允许特定路径？

OAUTH-接入

核心操作流程

进入 OAuth 配置页面

- 登录管理控制台。
- 左侧导航栏点击「系统设置」，进入系统配置页面。
- 在配置页面中找到「自定义 OAuth 提供商」模块，点击「添加 OAuth 提供商」按钮，弹出配置弹窗。

核心凭证配置

配置项	填写规则	示例
Client ID*	从 OAuth 提供商平台获取的客户端唯一标识	Iv1.xxxxxxxxxxxxxxxx
Client Secret*	从 OAuth 提供商平台获取的客户端密钥，严格保密，禁止泄露	GOCSPX-xxxxxxxxxxxxxxxxxxxx

OAuth 端点配置

配置项	填写规则	示例
授权端点 *	OAuth 提供商授权接口地址， 用于发起授权请求	https://g
令牌端点 *	用于兑换访问令牌的接口地址	https://g
用户信息端点 *	用于获取授权用户基本信息的接口地址	https://g

权限范围配置

Scopes（可选）：授权权限范围，多个 scope 用空格分隔，遵循 OAuth 提供商规范。
通用基础配置：openid profile email
自定义配置：根据提供商需求补充（如 GitHub 的repo、GitLab 的 read_user）

启用配置

配置完成后，可选择开启「启用供应商」开关，配置保存后立即生效；若暂不启用，可保持关闭状态。
点击「保存」按钮，完成 OAuth 提供商新增。

字段映射配置说明

用于配置系统如何从 OAuth 用户信息 API 的响应中，提取并映射为系统所需的用户数据，支持 JSONPath 语法，实现灵活的字段适配。

配置项 作用 示例值 说明
用户 ID 字段（可选） 用于唯一标识用户的字段路径，是 OAuth 账号与系统用户绑定的核心依据 sub 绝大多数标准 OAuth 服务会返回sub字段作为用户唯一标识，建议优先使用此字段
用户名字段（可选） 映射 OAuth 返回的用户名 / 账号字段 admin 可根据实际响应字段调整，如login、username、account等
显示名字段（可选） 映射 OAuth 返回的用户昵称 / 显示名称字段 username 可替换为name、nickname等，作为系统内用户的展示名称
邮箱字段（可选） 映射 OAuth 返回的用户邮箱字段 email 需确保 OAuth 服务已授权email scope，否则可能返回空值

```
{
  "sub": "12345678",
  "login": "zhangsan",
  "name": "张三",
  "email": "zhangsan@example.com",
  "profile": {
    "nickname": "小张"
  }
}
```

配置项	填写值	说明
用户 ID 字段	sub	直接取根节点的sub值
用户名字段	login	取根节点的login值
显示名字段	$.profile.nickname	使用 JSONPath

配置项	填写值	说明
		语法取嵌套节点的昵称
邮箱字段	email	取根节点的email值

参考

https://github.com/QuantumNous/new-api/issues/3334

AI平台与BIII对账（4-13）

模型价格计算：

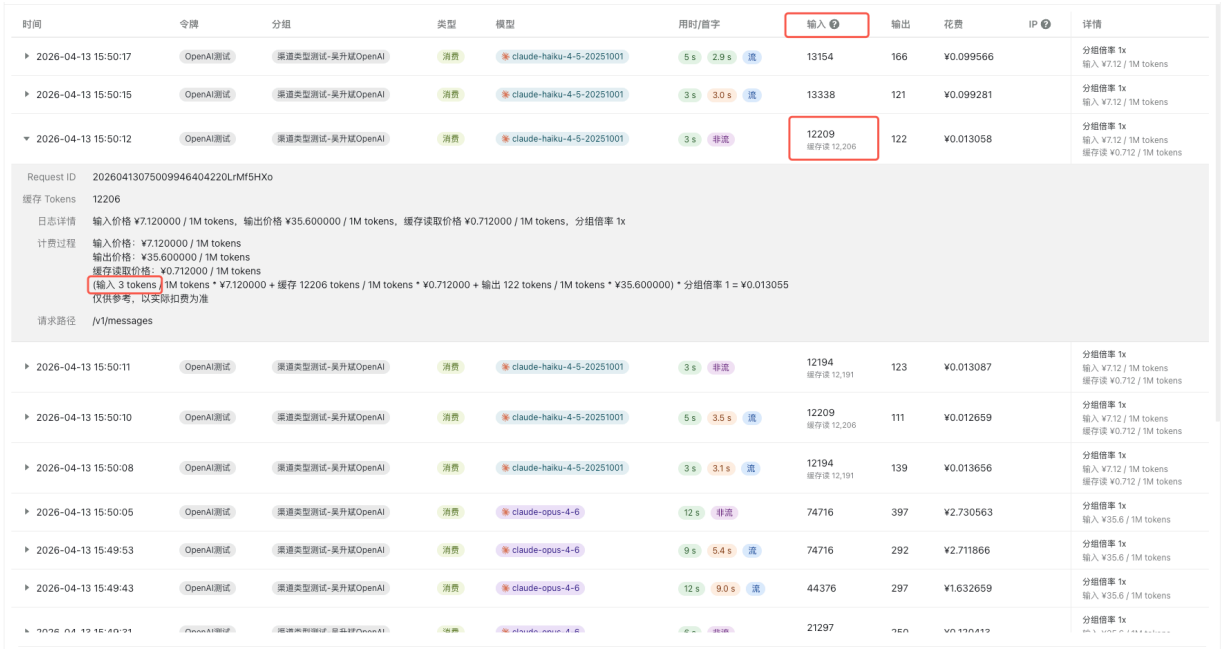
- 使用同一个模型 claude-opus-4-6
- 使用不同的提供商类型：Anthropic、OpenAI
- 与引用平台计算不同的价格

AI平台与BIII平台对账(使用OpenAI模型提供商，使用模型 claude-opus-4-6)

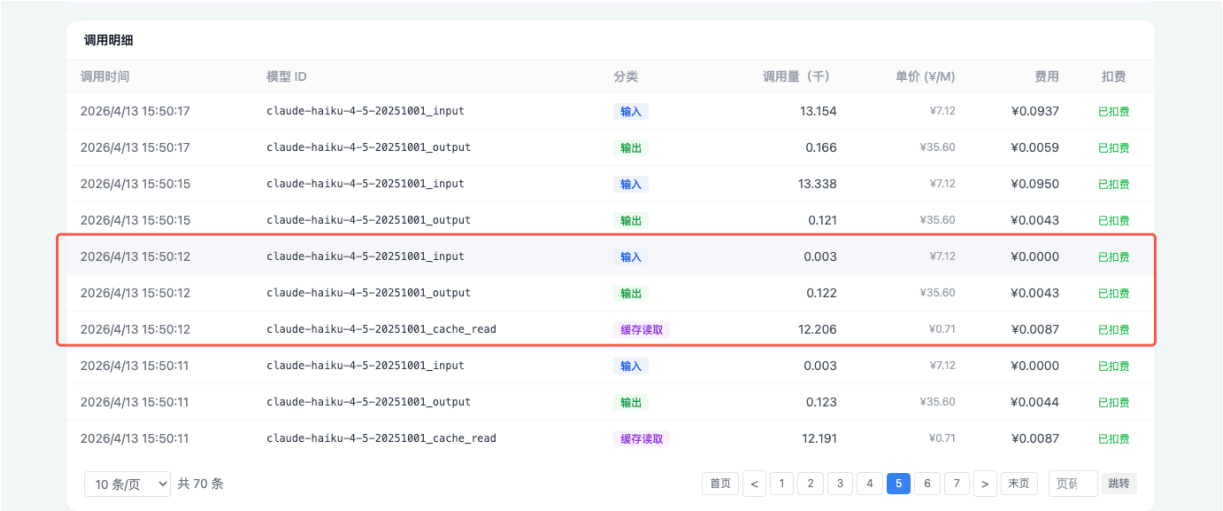
类型	AI平台	bill平台	差额
总金额	15.73	15.7294	0.0006
总token	7115590	7050142	65448
输入token	996818	905436	91382
输出token	20056	20056	0
缓存写入token	0	0	0
缓存读取token	91382	91382	0
请求次数	32	32	0

备注：存在6笔有缓存读取token总和与上述差额的持平（91382）

显示效果



时间	令牌	分组	类型	模型	用时/首字	输入	输出	花费	IP	详情
2026-04-13 15:50:17	OpenAI测试	渠道类型测试-吴升斌OpenAI	消费	claude-haiku-4-5-20251001	5 s 2.9 s 流	13154	166	¥0.099566		分组倍率 1x 输入 ¥7.12 / 1M tokens
2026-04-13 15:50:15	OpenAI测试	渠道类型测试-吴升斌OpenAI	消费	claude-haiku-4-5-20251001	3 s 3.0 s 流	13338	121	¥0.099281		分组倍率 1x 输入 ¥7.12 / 1M tokens
2026-04-13 15:50:12	OpenAI测试	渠道类型测试-吴升斌OpenAI	消费	claude-haiku-4-5-20251001	3 s 非流	12209 缓存读 12,206	122	¥0.013058		分组倍率 1x 输入 ¥7.12 / 1M tokens 缓存读 ¥0.712 / 1M tokens
Request ID	20260413075009946404220LrMf5HXo									
缓存 Tokens	12206									
日志详情	输入价格 ¥7.120000 / 1M tokens，输出价格 ¥35.600000 / 1M tokens，缓存读取价格 ¥0.712000 / 1M tokens，分组倍率 1x									
计费过程	输入价格：¥7.120000 / 1M tokens 输出价格：¥35.600000 / 1M tokens 缓存读取价格：¥0.712000 / 1M tokens (输入 3 tokens / 1M tokens * ¥7.120000 + 缓存 12206 tokens / 1M tokens * ¥0.712000 + 输出 122 tokens / 1M tokens * ¥35.600000) * 分组倍率 1 = ¥0.013055 仅供参考，以实际扣费为准									
请求路径	/v1/messages									
2026-04-13 15:50:11	OpenAI测试	渠道类型测试-吴升斌OpenAI	消费	claude-haiku-4-5-20251001	3 s 非流	12194 缓存读 12,191	123	¥0.013087		分组倍率 1x 输入 ¥7.12 / 1M tokens 缓存读 ¥0.712 / 1M tokens
2026-04-13 15:50:10	OpenAI测试	渠道类型测试-吴升斌OpenAI	消费	claude-haiku-4-5-20251001	5 s 3.5 s 流	12209 缓存读 12,206	111	¥0.012659		分组倍率 1x 输入 ¥7.12 / 1M tokens 缓存读 ¥0.712 / 1M tokens
2026-04-13 15:50:08	OpenAI测试	渠道类型测试-吴升斌OpenAI	消费	claude-haiku-4-5-20251001	3 s 3.1 s 流	12194 缓存读 12,191	139	¥0.013656		分组倍率 1x 输入 ¥7.12 / 1M tokens 缓存读 ¥0.712 / 1M tokens
2026-04-13 15:50:05	OpenAI测试	渠道类型测试-吴升斌OpenAI	消费	claude-opus-4-6	12 s 非流	74716	397	¥2.730563		分组倍率 1x 输入 ¥35.6 / 1M tokens
2026-04-13 15:49:53	OpenAI测试	渠道类型测试-吴升斌OpenAI	消费	claude-opus-4-6	9 s 5.4 s 流	74716	292	¥2.711866		分组倍率 1x 输入 ¥35.6 / 1M tokens
2026-04-13 15:49:43	OpenAI测试	渠道类型测试-吴升斌OpenAI	消费	claude-opus-4-6	12 s 9.0 s 流	44376	297	¥1.632659		分组倍率 1x 输入 ¥35.6 / 1M tokens
						21297				分组倍率 1x



调用明细

调用时间	模型 ID	分类	调用量（千）	单价 (¥/M)	费用	扣费
2026/4/13 15:50:17	claude-haiku-4-5-20251001_input	输入	13.154	¥7.12	¥0.0937	已扣费
2026/4/13 15:50:17	claude-haiku-4-5-20251001_output	输出	0.166	¥35.60	¥0.0059	已扣费
2026/4/13 15:50:15	claude-haiku-4-5-20251001_input	输入	13.338	¥7.12	¥0.0950	已扣费
2026/4/13 15:50:15	claude-haiku-4-5-20251001_output	输出	0.121	¥35.60	¥0.0043	已扣费
2026/4/13 15:50:12	claude-haiku-4-5-20251001_input	输入	0.003	¥7.12	¥0.0000	已扣费
2026/4/13 15:50:12	claude-haiku-4-5-20251001_output	输出	0.122	¥35.60	¥0.0043	已扣费
2026/4/13 15:50:12	claude-haiku-4-5-20251001_cache_read	缓存读取	12.206	¥0.71	¥0.0087	已扣费
2026/4/13 15:50:11	claude-haiku-4-5-20251001_input	输入	0.003	¥7.12	¥0.0000	已扣费
2026/4/13 15:50:11	claude-haiku-4-5-20251001_output	输出	0.123	¥35.60	¥0.0044	已扣费
2026/4/13 15:50:11	claude-haiku-4-5-20251001_cache_read	缓存读取	12.191	¥0.71	¥0.0087	已扣费

10 条/页 共 70 条　首页 < 1 2 3 4 5 6 7 > 末页　跳转

AI平台与BIII平台对账(使用Anthropic模型提供商，使用模型 claude-opus-4-6)

类型	AI平台	bill平台	差额
总金额	17.11	17.1080	0.0080
总token	2880911	2880911	0
输入token	9305	9305	0
输出token	8150	8150	0
缓存写入token	128967	128967	0
缓存读取token	2734489	2734489	0
请求次数	37	37	0

添加销售角色（4-13）

用户角色设计成销售解决方案

一、将用户设为管理员，控制菜单（推荐）

优势：

1、代码改动极小，开发成本低
2、直接复用现有权限体系，无缝兼容
3、上线快、风险低，不影响原有功能

缺点：

1、后台界面体验一般
2、无专属销售定制化界面
3、菜单权限控制粒度较粗，不够灵活

扩展：

1、增加开关配置，超级管理员可直接将普通用户设为管理员
2、可简单过滤部分敏感菜单，实现基础权限区分

二、新增销售管理员角色

优势：

1、角色职责清晰，权限完全独立隔离
2、支持菜单、接口、数据精细化权限控制
3、便于后续扩展客服、财务、审计等更多角色

缺点：

1、涉及角色、权限配置，代码改动量大
2、需要重新调整权限逻辑与菜单绑定
3、改动与测试周期长，上线风险略高

企业级邮箱接入（4-13）

核心操作流程

进入 OAuth 配置页面

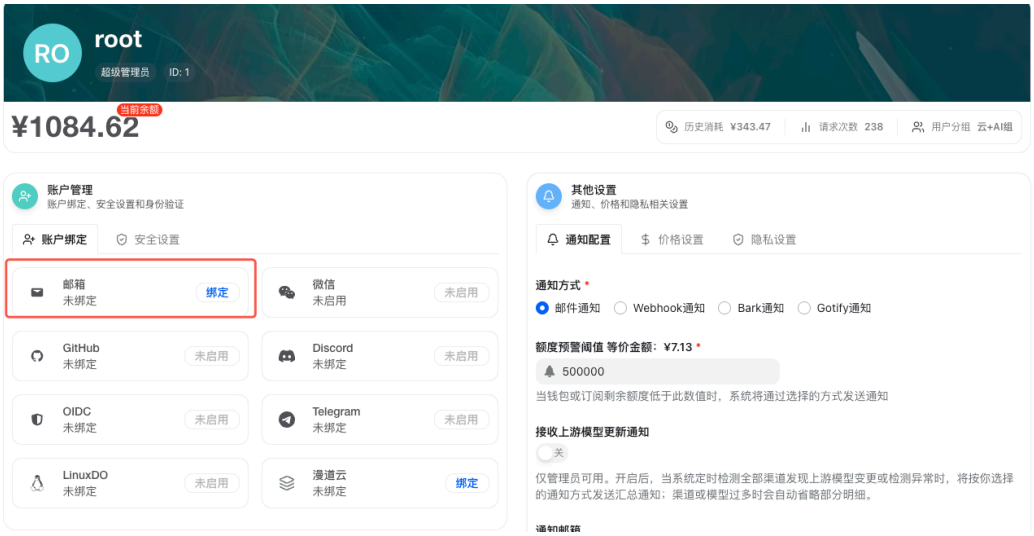
- 登录管理控制台。
- 左侧导航栏点击「系统设置」，进入系统配置页面。
- 在配置页面中找到「配置 SMTP」模块、配置服务邮箱。

用以支持系统的邮件发送

- SMTP 服务器地址 smtp.qiye.163.com
- SMTP 端口 465
- SMTP 账户 cloud@baofu.com
- SMTP 发送者邮箱 cloud@baofu.com
- SMTP 访问凭证 xxxx

  敏感信息不会发送到前端显示

用户可以绑定邮箱


RO
root
超级管理员
ID: 1
当前余额
¥1084.62
历史消耗 ¥343.47
请求次数 238
用户分组 云+AI组
账户管理
账户绑定、安全设置和身份验证
账户绑定
安全设置
邮箱
未绑定
绑定
微信
未启用
未启用
GitHub
未绑定
未启用
Discord
未绑定
未启用
OIDC
未绑定
未启用
Telegram
未绑定
未启用
LinuxDO
未绑定
未启用
漫道云
未绑定
绑定
其他设置
通知、价格和隐私相关设置
通知配置
价格设置
隐私设置
通知方式 *
邮件通知
Webhook通知
Bark通知
Gotify通知
额度预警阈值 等价金额：¥7.13 *
500000
当钱包或订阅剩余额度低于此数值时，系统将通过选择的方式发送通知
接收上游模型更新通知
关
仅管理员可用。开启后，当系统定时检测全部渠道发现上游模型变更或检测异常时，将按你选择的通知方式发送汇总通知；渠道或模型过多时会自动省略部分明细。

Ai-model-网址配置（4-13）

mandao域名配置

```
## 配置mandao【管理端和API端】页面地址配置-80端口
server {
    listen 80;
    server_name ai-api.mandao.com ai-admin.mandao.com;
    charset zh-CN;
    return 301 https://$host$request_uri;
}

## 配置mandao【管理端和API端】页面地址配置-443端口
server {
    listen 443 ssl;
    server_name ai-api.mandao.com ai-admin.mandao.com;
    ssl_certificate   /etc/nginx/conf.d/cert/mandao/mandao.pem;
    ssl_certificate_key  /etc/nginx/conf.d/cert/mandao/mandao.key;
    ssl_session_timeout 5m;
    ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE:ECDH:AES:HIGH:!NULL:!aNULL:!MD5:!ADH:!RC4;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers on;
    client_header_buffer_size 1m;
    large_client_header_buffers 4 10m;
```

```
    ## 登录跳转、短信发送
    location ~ ^/api/v1_1/(oauth/authorize|sms/send) {
       proxy_pass http://172.21.6.53$request_uri;
       proxy_set_header Host $host;
       proxy_set_header X-Real-IP $remote_addr;
       proxy_set_header X-Forwarded-For $proxy_add_x_forwa
rded_for;
       proxy_set_header X-Forwarded-Proto $scheme;
       proxy_redirect off;
    }

    ## 短信认证
    location /api/v1_1/sms/verify {
        proxy_pass http://172.21.6.53$request_uri;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw
arded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_redirect off;
    }

    location / {
       proxy_pass http://172.21.15.79$request_uri;
       proxy_set_header Host $host;
       proxy_set_header X-Real-IP $remote_addr;
       proxy_set_header X-Forwarded-For $proxy_add_x_forwa
rded_for;
```

```
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_redirect off;
    }
}
```

mandao-tech域名配置

```
## 配置mandao-tech【前端】页面地址配置-80端口
server {
    listen 80;
    server_name ai.mandao.com;
    charset zh-CN;
    return 301 https://$host$request_uri;
}

## 配置mandao-tech【前端】页面地址配置-443端口
server {
    listen 443 ssl;
    server_name ai.mandao.com;
    ssl_certificate   /etc/nginx/conf.d/cert/mandao/mandao.pem;
    ssl_certificate_key  /etc/nginx/conf.d/cert/mandao/mandao.key;
    ssl_session_timeout 5m;
    ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE:ECDH:AES:HIGH:!NULL:!aNULL:!MD5:!ADH:!RC4;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
```

```
    ssl_prefer_server_ciphers on;
    client_header_buffer_size 1m;
    large_client_header_buffers 4 10m;

    location / {
        proxy_pass http://172.21.9.58$request_uri;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw
arded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_redirect off;
    }
}
## 配置mandao-tech【管理端和API端】页面地址配置-80端口
server {
    listen 80;
    server_name ai-api.mandao-tech.com ai-admin.mandao-tec
h.com;
    charset zh-CN;
    return 301 https://$host$request_uri;
}

## 配置mandao-tech【管理端和API端】页面地址配置-443端口
server {
    listen 443 ssl;
    server_name ai-api.mandao-tech.com ai-admin.mandao-tec
h.com;
    ssl_certificate   /etc/nginx/conf.d/cert/mandao-tech/m
```

```
andao-tech.pem;
    ssl_certificate_key  /etc/nginx/conf.d/cert/mandao-tec
h/mandao-tech.key;
    ssl_session_timeout 5m;
    ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE:ECDH:AE
S:HIGH:!NULL:!aNULL:!MD5:!ADH:!RC4;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers on;
    client_header_buffer_size 1m;
    large_client_header_buffers 4 10m;

    ## 登录跳转、短信发送
    location ~ ^/api/v1_1/(oauth/authorize|sms/send) {
       proxy_pass http://172.21.6.53$request_uri;
       proxy_set_header Host $host;
       proxy_set_header X-Real-IP $remote_addr;
       proxy_set_header X-Forwarded-For $proxy_add_x_forwa
rded_for;
       proxy_set_header X-Forwarded-Proto $scheme;
       proxy_redirect off;
    }

    ## 短信认证
    location /api/v1_1/sms/verify {
        proxy_pass http://172.21.6.53$request_uri;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw
arded_for;
```

```
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_redirect off;
    }

    location / {
       proxy_pass http://172.21.15.79$request_uri;
       proxy_set_header Host $host;
       proxy_set_header X-Real-IP $remote_addr;
       proxy_set_header X-Forwarded-For $proxy_add_x_forwa
rded_for;
       proxy_set_header X-Forwarded-Proto $scheme;
       proxy_redirect off;
    }
}

## 配置mandao-tech【前端】页面地址配置-80端口
server {
    listen 80;
    server_name ai.mandao-tech.com;
    charset zh-CN;
    return 301 https://$host$request_uri;
}

## 配置mandao-tech【前端】页面地址配置-443端口
server {
    listen 443 ssl;
    server_name ai.mandao-tech.com;
    ssl_certificate   /etc/nginx/conf.d/cert/mandao-tech/m
andao-tech.pem;
```

```
    ssl_certificate_key  /etc/nginx/conf.d/cert/mandao-tech/mandao-tech.key;
    ssl_session_timeout 5m;
    ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE:ECDH:AES:HIGH:!NULL:!aNULL:!MD5:!ADH:!RC4;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers on;
    client_header_buffer_size 1m;
    large_client_header_buffers 4 10m;

    location / {
        proxy_pass http://172.21.9.58$request_uri;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_redirect off;
    }
}
```

AI-admin-替换主题（04-14）

Web端使用`semi`框架的自定义主题

一、进入主题编辑平台

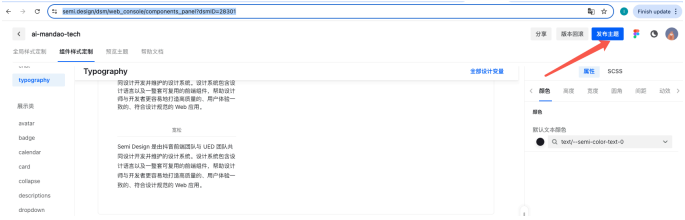
打开 Semi 官方 DSM 主题编辑器：
https://semi.design/dsm/web_console/components_panel?dsmID=28301

在平台内可自由修改：

- 主色 / 辅助色 / 功能色
- 圆角、阴影、间距
- 字体、字号
- 各组件样式（按钮、输入框、弹窗等）

二、编辑完成后发布主题

编辑完成后点击右上角 发布，系统会自动打包生成专属主题包。

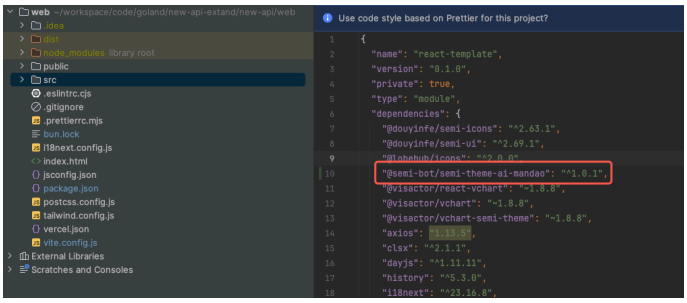


三、安装主题包

在项目 web 目录下执行：

```
bun add @semi-bot/semi-theme-ai-mandao
```

安装成功后，可在 package.json 中看到已安装：@semi-bot/semi-theme-ai-mandao

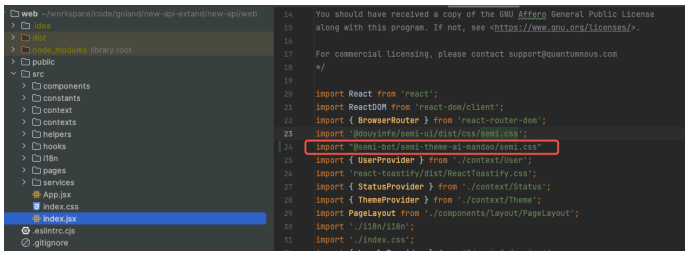


四、在项目中引入主题样式

打开文件：
web/src/index.jsx
按如下顺序引入样式（必须保证顺序）：

```
// 1. 先引入 Semi 官方基础样式
import '@douyinfe/semi-ui/dist/css/semi.css';

// 2. 再引入自定义主题（必须放在下一行，避免样式覆盖异常）
import '@semi-bot/semi-theme-ai-mandao/dist/css/semi.css';
```



> 注意
> 主题样式必须放在官方 semi.css 之后
> 确保不要重复引入、颠倒顺序

工作量统计

AI算力平台待解决问题
1、AI 平台使用请求限制用户请求。
2、用户已兑换券，管理端修改用户信息时，影响兑换券。
3、短信短时间多次发送。
4、API请求地址调整。
5、项目健康检查调整。

- 通知中心
- 授权服务
- AI服务端

  6、增加准生产配置。

04-07

- 增加 ai.mandao.com /ai-api.mandao.com域名 （完成）

  （ai.mandao-tech.com未备案）
- 项目健康检查调整。 完成
  - 通知中心
  - 授权服务
  - AI服务端

04-08

- 视频模型对比 （还需要找模型）
    - 阿里云
    - 腾讯
    - 火山
    - Ucloud
- 短信签名替换（宝付支付->上海漫道科技）

04-09

- 授权平台支持多域名短信登录 - 完成
- 统计不同模型提供商的价格计算（OpenAI、Anthropic）- 完成

04-10

- 编写 AI平台的功能列表
- 开发增加用户列表的创建时间

04-13

- AI算力使用手册文档
- 去掉多余额AI算力平台的301跳转
- 日志打印到控制台问题解决
- 企业级邮箱接入

04-16

- 完成前后端代码分离
- 编写前后端的dockerfile
- 去除多余的项目代码
- 去除项目中多余的非法字符

未完成事项

- 对接企业级用户
- mandao-tech.com ICP备案认证 （申请当中）
- AI服务端增加用户创建时间
- 平台支持oauth标准登录
- 增加企业级角色 (改动大,涉及 使用量、用户组)

需要控制的菜单权限

左侧与顶部菜单功能控制说明

顶部导航

显示控制：web/src/hooks/common/useNavigation.js 第 36 行
权限控制：web/src/App.jsx

- 首页、模型广场、关于

  不显示

- 数据看板、使用日志、任务日志

  正常显示

用户管理界面

- 表格操作按钮：提升、降级、订阅管理、重置 Passkey、控制 2FA、注销

  文件路径：

  web/src/components/table/users/UsersColumnDefs.jsx
- 编辑弹窗及用户绑定信息

  文件路径：

web/src/components/table/users/modals/EditUserModal.jsx

渠道管理

- 渠道列表展示

  文件路径：

  web/src/components/table/channels/ChannelsTable.jsx
- 渠道编辑（仅允许修改 “使用模型”）

  文件路径：

  web/src/components/table/channels/modals/EditChannelModal.jsx

渠道模型

- 模型列表（仅查看权限，不可编辑）

  文件路径：web/src/components/table/models/ModelsTable.jsx

方案

服务端增加一个 销售管理员角色：11，进行权限控制。
文件路径：web/src/helpers/utils.jsx

权限方案2

RBAC 权限系统方案 — 最小改动版

现状分析

当前系统的权限控制非常简单：

角色	值	状态
Guest	0	占位，未使用
Common	1	普通用户
Admin	10	管理员
custom	11	已定义，未实现
Root	100	超级管理员

权限判断只有一种方式：`role >= 阈值`。没有细粒度控制——要么看到整个管理区，要么完全看不到。

最小改动方案：复用 options 表 + 内存缓存

核心思路

在现有 `options` 表中新增一个 key `RolePermissions` ，存 JSON 格式的角色权限矩阵。不建新表、不改用户表结构。

完全复用 `SidebarModulesAdmin` 的模式：options 表存储 → `OptionMap` 内存缓存 → `/api/status` 下发给前端。

存储格式

```
{
  "10": ["menu:admin.channel", "menu:admin.models", "channel:create", "channel:update", "..."],
  "11": ["menu:admin.redemption", "menu:admin.user", "redemption:create", "..."],
  "1":  ["menu:chat.playground", "menu:console.token", "token:create", "..."],
  "100": ["*"]
}
```

- key 是角色值的字符串，value 是该角色拥有的权限数组
- Root 用 `["*"]` 代表全部权限
- Guest 无条目

权限命名规则：`资源:动作`

菜单权限（控制页面/菜单可见性）：

```
menu:chat.playground    menu:chat.chat
menu:console.detail     menu:console.model_usage    menu:console.token
menu:console.log        menu:console.midjourney     menu:console.task
menu:personal.topup     menu:personal.personal
menu:admin.channel      menu:admin.models           menu:admin.deployment
menu:admin.subscription menu:admin.redemption       menu:admin.user
menu:admin.setting
```

操作权限（控制页面内按钮/动作）：

```
channel:create   channel:update   channel:delete   channel:test
token:create     token:update     token:delete
user:create      user:update      user:delete      user:set_role
redemption:create  redemption:update  redemption:delete
model:create     model:update     model:delete
setting:update   log:delete
```

各角色默认权限矩阵

权限类别	Common(
menu:chat.* / console.* / personal.*	全部

权限类别	Common(
menu:admin.channel/models/deployment/subscription	-
menu:admin.redemption/user	-
menu:admin.setting	-
channel 增删改测	-
token 增删改	全部
user 增删改	-
redemption 增删改	-
setting:update / log:delete	-

改动清单

后端（4 个文件新增，3 个文件修改）

操作	文件	说明
新增	`common/permissions.go`	权限常量定义 + `DefaultRolePermiss` 返回默认矩阵 + 从 Op

操作	文件	说明
		解析的内存缓存 `HasP` `perm)`
修改	`model/option.go`	`InitOptionMap` 加入 `RolePermissions` 默认值；`updateOpti` 解析 JSON 到内存缓存
新增	`middleware/permission.go`	`RequirePermission(p` 中间件，从缓存查权限
新增	`controller/role_permission.go`	Root-only API：查询/更 重置角色权限（读写 c `RolePermissions` ke
修改	`router/api-router.go`	各路由组追加 `Requir` 兑换码路由从 `AdminA` `UserAuth + RequirePermission` 加权限管理路由
修改	`controller/user.go`	`GetSelf` / `setupLog` `permissions` 字段 （从缓存读当前角色权
修改	`controller/misc.go`	`GetStatus` 返回 `RolePermissions` （

前端（3 个文件新增，5+ 个文件修改）

操作	文件	说
新增	`helpers/permission.js`	`I` `I` 工
新增	`hooks/common/usePermission.js`	R
新增	`pages/Setting/System/RolePermissionSettings.jsx`	R 按
修改	`helpers/auth.jsx`	新
修改	`App.jsx`	A `<` pe
修改	`hooks/common/useSidebar.js`	三 Al
修改	`components/layout/SiderBar.jsx`	`I`
修改	各 Actions 组件	按 `I` 才

i18n（7 个文件修改）

所有 `web/src/i18n/locales/*.json` 添加权限名称和权限管理页面的翻译。

数据流

```
启动 → InitOptionMap 写入默认 RolePermissions → 内存缓存就绪
                                                    ↓
用户登录 → setupLogin → 从缓存读角色权限 → 返回 { permissions: [...] }
                                                    ↓
前端存 localStorage → useSidebar/usePermission 读取 → 菜单和按钮按权限显示
                                                    ↓
API 请求 → UserAuth/AdminAuth（第一道门槛）→ RequirePermission（第二道门槛）
                                                    ↓
Root 修改权限 → UpdateOption("RolePermissions", json) → 刷新缓存
                    → 用户下次刷新页面自动获取新权限
```

向后兼容

- 不改 users 表，`role` 字段不变
- 保留 `UserAuth` / `AdminAuth` / `RootAuth` 作为第一道门槛
- `RequirePermission` 作为叠加的第二道门槛
- 前端检测不到 `permissions` 时回退到原有 `isAdmin()` / `isRoot()` 逻辑
- 首次启动自动写入默认权限，现有用户无感迁移

总结

- 不新增数据库表，复用 options 表存一条 JSON
- 不改用户表结构
- 后端新增 3 个文件、修改 4 个文件
- 前端新增 3 个文件、修改 5-8 个文件
- 完全兼容现有逻辑，渐进式叠加

权限方案3

权限设计3

资源定义

```
{
  "menu-home": "首页",
  "menu-cluster": "集群管理",

  "home-btn_add": "首页-添加按钮",
  "home-btn_delete": "首页-删除按钮",

  "cluster-create_btn": "创建集群",
  "cluster-delete_btn": "删除集群"

...
  # 格式
  # menu-xxx          → 菜单
  # xxx-btn_xxx       → 页面按钮
  # xxx-tab_xxx       → tab
  # xxx-field_xxx     → 字段
}
```

权限分配结构（核心）

域名（host）

```
{
  "admin.example.com":
  {
    "menu-home": ""
  }
}
用户组（group）
用户（user）
```

{
“host”: {
“admin.example.com”: {
“allow”: [“menu-home”, “menu-cluster”],
“deny”: []
}
},

```
"group": {
  "dev": {
    "allow": [
      "menu-home",
      "home-add_btn",
      "cluster-create_btn"
    ],
```

```
    "deny": [
      "cluster-delete_btn"
    ]
  }
},

"user": {
  "lance": {
    "allow": ["home-delete_btn"],
    "deny": []
  }
}
```

}

`

日志错误分析

日志错误类型统计与分析

统计范围：2026-04-17 ~ 2026-04-20 全部 ERR 日志，去重按错误类型归类，统计出现频次

一、错误类型 & 笔数统计

- 客户端主动断开 / 上下文取消（client_gone /context canceled）

  笔数：112 笔

  典型日志：

  stream ended: reason=client_gone end_error="context canceled"

  附带：error handling last response: unexpected end of JSON input

  说明：用户 / 客户端主动取消请求、断连、超时关闭连接。

- 模型无可用渠道 / 无 distributor（Claude/GPT 系列为主）

  笔数：33 笔

  典型日志：

  分组 default 下模型 claude-opus-4-7 无可用渠道（distributor）

  模型 无可用渠道

  说明：后端未配置该模型的转发渠道、渠道下线或负载不可用。

- 用户余额 / 额度不足（预扣费失败）

  笔数：12 笔

  典型日志：

  预扣费额度失败, 用户剩余额度: xxx, 需要: xxx

  用户额度不足, 剩余额度: ￥-3.152380

  说明：账户余额不足以支付本次模型调用费用。

- temperature 参数已废弃（Bedrock Claude 模型）

  笔数：9 笔

  典型日志：

  temperature is deprecated for this model

  说明：请求携带了该模型已不支持的 temperature 参数，参数非法。

claude-opus-4-7 时传了 temperature，而这个模型明确禁止使用 temperature，所以返回 400 错误。

- 上游服务 504 超时（图像生成）

  笔数：1 笔

  日志：图像生成超时（5 分钟）

  说明：wan2.7-image 模型生成图片超时。

- Token 权限不足 / 无权访问模型

  笔数：7 笔

典型：该令牌无权访问模型 claude-haiku-4-5-20251001

- Token 额度超限（429）

  笔数：2 笔

  日志：Token limit exceeded: 500337885 / 500000000

- 上游连接超时 / 请求失败（500）

  笔数：2 笔

  日志：read: connection timed out、upstream error: do request failed

- 未提供令牌 / 无效令牌

  笔数：3 笔

  日志：未提供令牌、无效的令牌

- 代理错误 502

  笔数：1 笔

- 操作不支持（Invalid param）

  笔数：1 笔

  日志：The requested operation is unsupported

二、整体汇总

总错误条数：174 条
占比最高三类：
客户端断开：64.4%
模型无可用渠道：18.9%
余额不足：6.9%

三、问题结论

最主要问题：大量用户中途取消 / 断连，导致流异常结束。
平台侧问题：Claude/GPT 系列渠道缺失 / 未配置是第二大原因。
业务问题：较多用户余额不足，影响可用性。
参数兼容问题：部分请求带了已废弃的 temperature，需前端 / SDK 屏蔽。
少量网络 / 上游超时：集中在图片生成与远端服务连接。

Ai-hub数据库表结构

能力表

```
SET NAMES utf8mb4;
SET FOREIGN_KEY_CHECKS = 0;

-- ----------------------------
-- Table structure for abilitie
s
-- ----------------------------
DROP TABLE IF EXISTS `abilities
`;
CREATE TABLE `abilities` (
  `group` varchar(64) COLLATE u
tf8mb4_unicode_ci NOT NULL,
  `model` varchar(255) COLLATE
utf8mb4_unicode_ci NOT NULL,
  `channel_id` bigint NOT NULL,
  `enabled` tinyint(1) DEFAULT
NULL,
  `priority` bigint DEFAULT
'0',
  `weight` bigint unsigned DEFA
ULT '0',
```

```
  `tag` varchar(191) COLLATE ut
f8mb4_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`group`,`model`,
`channel_id`),
  KEY `idx_abilities_channel_id
` (`channel_id`),
  KEY `idx_abilities_priority`
(`priority`),
  KEY `idx_abilities_weight` (`
weight`),
  KEY `idx_abilities_tag` (`tag
`)
) ENGINE=InnoDB DEFAULT CHARSET
=utf8mb4 COLLATE=utf8mb4_unicod
e_ci;
```

渠道表

```
-- ----------------------------
-- Table structure for channels
-- ----------------------------
DROP TABLE IF EXISTS `channels
`;
CREATE TABLE `channels` (
  `id` bigint NOT NULL AUTO_INC
REMENT,
  `type` bigint DEFAULT '0',
```

```
  `key` longtext COLLATE utf8mb
4_unicode_ci NOT NULL,
  `open_ai_organization` longte
xt COLLATE utf8mb4_unicode_ci,
  `test_model` longtext COLLATE
utf8mb4_unicode_ci,
  `status` bigint DEFAULT '1',
  `name` varchar(191) COLLATE u
tf8mb4_unicode_ci DEFAULT NULL,
  `weight` bigint unsigned DEFA
ULT '0',
  `created_time` bigint DEFAULT
NULL,
  `test_time` bigint DEFAULT NU
LL,
  `response_time` bigint DEFAUL
T NULL,
  `base_url` varchar(191) COLLA
TE utf8mb4_unicode_ci DEFAULT
'',
  `other` longtext COLLATE utf8
mb4_unicode_ci,
  `balance` double DEFAULT NUL
L,
  `balance_updated_time` bigint
DEFAULT NULL,
  `models` longtext COLLATE utf
8mb4_unicode_ci,
  `group` varchar(64) COLLATE u
```

```
tf8mb4_unicode_ci DEFAULT 'defa
ult',
  `used_quota` bigint DEFAULT
'0',
  `model_mapping` text COLLATE
utf8mb4_unicode_ci,
  `status_code_mapping` varchar
(1024) COLLATE utf8mb4_unicode_
ci DEFAULT '',
  `priority` bigint DEFAULT
'0',
  `auto_ban` bigint DEFAULT
'1',
  `other_info` longtext COLLATE
utf8mb4_unicode_ci,
  `tag` varchar(191) COLLATE ut
f8mb4_unicode_ci DEFAULT NULL,
  `setting` text COLLATE utf8mb
4_unicode_ci,
  `param_override` text COLLATE
utf8mb4_unicode_ci,
  `header_override` text COLLAT
E utf8mb4_unicode_ci,
  `remark` varchar(255) COLLATE
utf8mb4_unicode_ci DEFAULT NUL
L,
  `channel_info` json DEFAULT N
ULL,
  `settings` longtext COLLATE u
```

```
tf8mb4_unicode_ci,
  PRIMARY KEY (`id`),
  KEY `idx_channels_name` (`nam
e`),
  KEY `idx_channels_tag` (`tag
`)
) ENGINE=InnoDB AUTO_INCREMENT=
22 DEFAULT CHARSET=utf8mb4 COLL
ATE=utf8mb4_unicode_ci;
```

```
-- ----------------------------
-- Table structure for checkins
-- ----------------------------
DROP TABLE IF EXISTS `checkins
`;
CREATE TABLE `checkins` (
  `id` bigint NOT NULL AUTO_INC
REMENT,
  `user_id` bigint NOT NULL,
  `checkin_date` varchar(10) CO
LLATE utf8mb4_unicode_ci NOT NU
LL,
  `quota_awarded` bigint NOT NU
LL,
  `created_at` bigint DEFAULT N
```

```
ULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `idx_user_checkin_
date` (`user_id`,`checkin_date
`)
) ENGINE=InnoDB DEFAULT CHARSET
=utf8mb4 COLLATE=utf8mb4_unicod
e_ci;
```

OAuth提供商表

```
-- ----------------------------
-- Table structure for custom_o
auth_providers
-- ----------------------------
DROP TABLE IF EXISTS `custom_oa
uth_providers`;
CREATE TABLE `custom_oauth_prov
iders` (
  `id` bigint NOT NULL AUTO_INC
REMENT,
  `name` varchar(64) COLLATE ut
f8mb4_unicode_ci NOT NULL,
  `slug` varchar(64) COLLATE ut
f8mb4_unicode_ci NOT NULL,
  `icon` varchar(128) COLLATE u
tf8mb4_unicode_ci DEFAULT '',
```

```
  `enabled` tinyint(1) DEFAULT
'0',
  `client_id` varchar(256) COLL
ATE utf8mb4_unicode_ci DEFAULT
NULL,
  `client_secret` varchar(512)
COLLATE utf8mb4_unicode_ci DEFA
ULT NULL,
  `authorization_endpoint` varc
har(512) COLLATE utf8mb4_unicod
e_ci DEFAULT NULL,
  `token_endpoint` varchar(512)
COLLATE utf8mb4_unicode_ci DEFA
ULT NULL,
  `user_info_endpoint` varchar
(512) COLLATE utf8mb4_unicode_c
i DEFAULT NULL,
  `scopes` varchar(256) COLLATE
utf8mb4_unicode_ci DEFAULT 'ope
nid profile email',
  `user_id_field` varchar(128)
COLLATE utf8mb4_unicode_ci DEFA
ULT 'sub',
  `username_field` varchar(128)
COLLATE utf8mb4_unicode_ci DEFA
ULT 'preferred_username',
  `display_name_field` varchar
(128) COLLATE utf8mb4_unicode_c
i DEFAULT 'name',
```

```
  `email_field` varchar(128) CO
LLATE utf8mb4_unicode_ci DEFAUL
T 'email',
  `well_known` varchar(512) COL
LATE utf8mb4_unicode_ci DEFAULT
NULL,
  `auth_style` bigint DEFAULT
'0',
  `access_policy` text COLLATE
utf8mb4_unicode_ci,
  `access_denied_message` varch
ar(512) COLLATE utf8mb4_unicode
_ci DEFAULT NULL,
  `created_at` datetime(3) DEFA
ULT NULL,
  `updated_at` datetime(3) DEFA
ULT NULL,
  `phone_field` varchar(128) CO
LLATE utf8mb4_unicode_ci DEFAUL
T 'phone',
  PRIMARY KEY (`id`),
  UNIQUE KEY `idx_custom_oauth_
providers_slug` (`slug`)
) ENGINE=InnoDB AUTO_INCREMENT=
2 DEFAULT CHARSET=utf8mb4 COLLA
TE=utf8mb4_unicode_ci;
```

日志表（使用量、充值记录、错误记录）

```
-- ----------------------------
-- Table structure for logs
-- ----------------------------
DROP TABLE IF EXISTS `logs`;
CREATE TABLE `logs` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `user_id` bigint DEFAULT NULL,
  `created_at` bigint DEFAULT NULL,
  `type` bigint DEFAULT NULL,
  `content` longtext COLLATE utf8mb4_unicode_ci,
  `username` varchar(191) COLLATE utf8mb4_unicode_ci DEFAULT '',
  `token_name` varchar(191) COLLATE utf8mb4_unicode_ci DEFAULT '',
  `model_name` varchar(191) COLLATE utf8mb4_unicode_ci DEFAULT '',
  `quota` bigint DEFAULT '0',
  `prompt_tokens` bigint DEFAUL
```

```
T '0',
  `completion_tokens` bigint DE
FAULT '0',
  `use_time` bigint DEFAULT
'0',
  `is_stream` tinyint(1) DEFAUL
T NULL,
  `channel_id` bigint DEFAULT N
ULL,
  `channel_name` longtext COLLA
TE utf8mb4_unicode_ci,
  `token_id` bigint DEFAULT
'0',
  `group` varchar(191) COLLATE
utf8mb4_unicode_ci DEFAULT NUL
L,
  `ip` varchar(191) COLLATE utf
8mb4_unicode_ci DEFAULT '',
  `request_id` varchar(64) COLL
ATE utf8mb4_unicode_ci DEFAULT
'',
  `other` longtext COLLATE utf8
mb4_unicode_ci,
  PRIMARY KEY (`id`),
  KEY `index_username_model_nam
e` (`model_name`,`username`),
  KEY `idx_logs_token_name` (`t
oken_name`),
  KEY `idx_logs_model_name` (`m
```

```
odel_name`),
  KEY `idx_logs_token_id` (`tok
en_id`),
  KEY `idx_logs_group` (`group
`),
  KEY `idx_logs_ip` (`ip`),
  KEY `idx_logs_request_id` (`r
equest_id`),
  KEY `idx_created_at_id` (`id
`,`created_at`),
  KEY `idx_user_id_id` (`user_i
d`,`id`),
  KEY `idx_logs_user_id` (`user
_id`),
  KEY `idx_created_at_type` (`c
reated_at`,`type`),
  KEY `idx_logs_username` (`use
rname`),
  KEY `idx_logs_channel_id` (`c
hannel_id`)
) ENGINE=InnoDB AUTO_INCREMENT=
39880 DEFAULT CHARSET=utf8mb4 C
OLLATE=utf8mb4_unicode_ci;
```

绘图表

```
-- ----------------------------
-- Table structure for midjourn
eys
-- ----------------------------
DROP TABLE IF EXISTS `midjourne
ys`;
CREATE TABLE `midjourneys` (
  `id` bigint NOT NULL AUTO_INC
REMENT,
  `code` bigint DEFAULT NULL,
  `user_id` bigint DEFAULT NUL
L,
  `action` varchar(40) COLLATE
utf8mb4_unicode_ci DEFAULT NUL
L,
  `mj_id` varchar(191) COLLATE
utf8mb4_unicode_ci DEFAULT NUL
L,
  `prompt` longtext COLLATE utf
8mb4_unicode_ci,
  `prompt_en` longtext COLLATE
utf8mb4_unicode_ci,
  `description` longtext COLLAT
E utf8mb4_unicode_ci,
  `state` longtext COLLATE utf8
mb4_unicode_ci,
  `submit_time` bigint DEFAULT
NULL,
  `start_time` bigint DEFAULT N
```

```
ULL,
  `finish_time` bigint DEFAULT
NULL,
  `image_url` longtext COLLATE
utf8mb4_unicode_ci,
  `video_url` longtext COLLATE
utf8mb4_unicode_ci,
  `video_urls` longtext COLLATE
utf8mb4_unicode_ci,
  `status` varchar(20) COLLATE
utf8mb4_unicode_ci DEFAULT NUL
L,
  `progress` varchar(30) COLLAT
E utf8mb4_unicode_ci DEFAULT NU
LL,
  `fail_reason` longtext COLLAT
E utf8mb4_unicode_ci,
  `channel_id` bigint DEFAULT N
ULL,
  `quota` bigint DEFAULT NULL,
  `buttons` longtext COLLATE ut
f8mb4_unicode_ci,
  `properties` longtext COLLATE
utf8mb4_unicode_ci,
  PRIMARY KEY (`id`),
  KEY `idx_midjourneys_progress
` (`progress`),
  KEY `idx_midjourneys_user_id`
(`user_id`),
```

```
  KEY `idx_midjourneys_action`
(`action`),
  KEY `idx_midjourneys_mj_id`
(`mj_id`),
  KEY `idx_midjourneys_submit_t
ime` (`submit_time`),
  KEY `idx_midjourneys_start_ti
me` (`start_time`),
  KEY `idx_midjourneys_finish_t
ime` (`finish_time`),
  KEY `idx_midjourneys_status`
(`status`)
) ENGINE=InnoDB DEFAULT CHARSET
=utf8mb4 COLLATE=utf8mb4_unicod
e_ci;
```

模型表

```
-- ----------------------------
-- Table structure for models
-- ----------------------------
DROP TABLE IF EXISTS `models`;
CREATE TABLE `models` (
  `id` bigint NOT NULL AUTO_INC
REMENT,
  `model_name` varchar(128) COL
LATE utf8mb4_unicode_ci NOT NUL
```

```
L,
  `description` text COLLATE ut
f8mb4_unicode_ci,
  `icon` varchar(128) COLLATE u
tf8mb4_unicode_ci DEFAULT NULL,
  `tags` varchar(255) COLLATE u
tf8mb4_unicode_ci DEFAULT NULL,
  `vendor_id` bigint DEFAULT NU
LL,
  `endpoints` text COLLATE utf8
mb4_unicode_ci,
  `status` bigint DEFAULT '1',
  `sync_official` bigint DEFAUL
T '1',
  `created_time` bigint DEFAULT
NULL,
  `updated_time` bigint DEFAULT
NULL,
  `deleted_at` datetime(3) DEFA
ULT NULL,
  `name_rule` bigint DEFAULT
'0',
  PRIMARY KEY (`id`),
  UNIQUE KEY `uk_model_name_del
ete_at` (`model_name`,`deleted_
at`),
  KEY `idx_models_vendor_id` (`
vendor_id`),
  KEY `idx_models_deleted_at`
```

```
(`deleted_at`)
) ENGINE=InnoDB AUTO_INCREMENT=
44 DEFAULT CHARSET=utf8mb4 COLL
ATE=utf8mb4_unicode_ci;
```

系统表

```
-- ----------------------------
-- Table structure for options
-- ----------------------------
DROP TABLE IF EXISTS `options`;
CREATE TABLE `options` (
  `key` varchar(191) COLLATE ut
f8mb4_unicode_ci NOT NULL,
  `value` longtext COLLATE utf8
mb4_unicode_ci,
  PRIMARY KEY (`key`)
) ENGINE=InnoDB DEFAULT CHARSET
=utf8mb4 COLLATE=utf8mb4_unicod
e_ci;
```

指纹表

```
-- ----------------------------
-- Table structure for passkey_
```

```
credentials
-- ----------------------------
DROP TABLE IF EXISTS `passkey_credentials`;
CREATE TABLE `passkey_credentials` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `user_id` bigint NOT NULL,
  `credential_id` varchar(512) COLLATE utf8mb4_unicode_ci NOT NULL,
  `public_key` text COLLATE utf8mb4_unicode_ci NOT NULL,
  `attestation_type` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `aa_guid` varchar(512) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `sign_count` int unsigned DEFAULT '0',
  `clone_warning` tinyint(1) DEFAULT NULL,
  `user_present` tinyint(1) DEFAULT NULL,
  `user_verified` tinyint(1) DEFAULT NULL,
  `backup_eligible` tinyint(1)
```

```
DEFAULT NULL,
  `backup_state` tinyint(1) DEF
AULT NULL,
  `transports` text COLLATE utf
8mb4_unicode_ci,
  `attachment` varchar(32) COLL
ATE utf8mb4_unicode_ci DEFAULT
NULL,
  `last_used_at` datetime(3) DE
FAULT NULL,
  `created_at` datetime(3) DEFA
ULT NULL,
  `updated_at` datetime(3) DEFA
ULT NULL,
  `deleted_at` datetime(3) DEFA
ULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `idx_passkey_crede
ntials_credential_id` (`credent
ial_id`),
  UNIQUE KEY `idx_passkey_crede
ntials_user_id` (`user_id`),
  KEY `idx_passkey_credentials_
deleted_at` (`deleted_at`)
) ENGINE=InnoDB DEFAULT CHARSET
=utf8mb4 COLLATE=utf8mb4_unicod
e_ci;
```

预填组表

```
-- ----------------------------
-- Table structure for prefill_groups
-- ----------------------------
DROP TABLE IF EXISTS `prefill_groups`;
CREATE TABLE `prefill_groups` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `name` varchar(64) COLLATE utf8mb4_unicode_ci NOT NULL,
  `type` varchar(32) COLLATE utf8mb4_unicode_ci NOT NULL,
  `items` json DEFAULT NULL,
  `description` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `created_time` bigint DEFAULT NULL,
  `updated_time` bigint DEFAULT NULL,
  `deleted_at` datetime(3) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `uk_prefill_name` (`name`),
```

```
  KEY `idx_prefill_groups_type` (`type`),
  KEY `idx_prefill_groups_deleted_at` (`deleted_at`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

限额表

```
-- ----------------------------
-- Table structure for quota_data
-- ----------------------------
DROP TABLE IF EXISTS `quota_data`;
CREATE TABLE `quota_data` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `user_id` bigint DEFAULT NULL,
  `username` varchar(64) COLLATE utf8mb4_unicode_ci DEFAULT '',
  `model_name` varchar(64) COLLATE utf8mb4_unicode_ci DEFAULT '',
```

```
  `created_at` bigint DEFAULT N
ULL,
  `token_used` bigint DEFAULT
'0',
  `count` bigint DEFAULT '0',
  `quota` bigint DEFAULT '0',
  PRIMARY KEY (`id`),
  KEY `idx_quota_data_user_id`
(`user_id`),
  KEY `idx_qdt_model_user_name`
(`model_name`,`username`),
  KEY `idx_qdt_created_at` (`cr
eated_at`)
) ENGINE=InnoDB AUTO_INCREMENT=
1410 DEFAULT CHARSET=utf8mb4 CO
LLATE=utf8mb4_unicode_ci;
```

兑换券表

```
-- ----------------------------
-- Table structure for redempti
ons
-- ----------------------------
DROP TABLE IF EXISTS `redemptio
ns`;
CREATE TABLE `redemptions` (
  `id` bigint NOT NULL AUTO_INC
```

```
REMENT,
  `user_id` bigint DEFAULT NUL
L,
  `key` char(32) COLLATE utf8mb
4_unicode_ci DEFAULT NULL,
  `status` bigint DEFAULT '1',
  `name` varchar(191) COLLATE u
tf8mb4_unicode_ci DEFAULT NULL,
  `quota` bigint DEFAULT '100',
  `created_time` bigint DEFAULT
NULL,
  `redeemed_time` bigint DEFAUL
T NULL,
  `used_user_id` bigint DEFAULT
NULL,
  `deleted_at` datetime(3) DEFA
ULT NULL,
  `expired_time` bigint DEFAULT
NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `idx_redemptions_k
ey` (`key`),
  KEY `idx_redemptions_name` (`
name`),
  KEY `idx_redemptions_deleted_
at` (`deleted_at`)
) ENGINE=InnoDB AUTO_INCREMENT=
```

```
56 DEFAULT CHARSET=utf8mb4 COLL
ATE=utf8mb4_unicode_ci;
```

系统初始化安装表

```
-- ----------------------------
-- Table structure for setups
-- ----------------------------
DROP TABLE IF EXISTS `setups`;
CREATE TABLE `setups` (
  `id` bigint unsigned NOT NULL
AUTO_INCREMENT,
  `version` varchar(50) COLLATE
utf8mb4_unicode_ci NOT NULL,
  `initialized_at` bigint NOT N
ULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=
2 DEFAULT CHARSET=utf8mb4 COLLA
TE=utf8mb4_unicode_ci;
```

订阅订单表表

```
-- ----------------------------
-- Table structure for subscrip
```

```
tion_orders
-- ----------------------------
DROP TABLE IF EXISTS `subscript
ion_orders`;
CREATE TABLE `subscription_orde
rs` (
  `id` bigint NOT NULL AUTO_INC
REMENT,
  `user_id` bigint DEFAULT NUL
L,
  `plan_id` bigint DEFAULT NUL
L,
  `money` double DEFAULT NULL,
  `trade_no` varchar(255) COLLA
TE utf8mb4_unicode_ci DEFAULT N
ULL,
  `payment_method` varchar(50)
COLLATE utf8mb4_unicode_ci DEFA
ULT NULL,
  `status` longtext COLLATE utf
8mb4_unicode_ci,
  `create_time` bigint DEFAULT
NULL,
  `complete_time` bigint DEFAUL
T NULL,
  `provider_payload` text COLLA
TE utf8mb4_unicode_ci,
  PRIMARY KEY (`id`),
  UNIQUE KEY `trade_no` (`trade
```

```
_no`),
  KEY `idx_subscription_orders_
user_id` (`user_id`),
  KEY `idx_subscription_orders_
plan_id` (`plan_id`),
  KEY `idx_subscription_orders_
trade_no` (`trade_no`)
) ENGINE=InnoDB DEFAULT CHARSET
=utf8mb4 COLLATE=utf8mb4_unicod
e_ci;
```

订阅计划表

```
-- ----------------------------
-- Table structure for subscrip
tion_plans
-- ----------------------------
DROP TABLE IF EXISTS `subscript
ion_plans`;
CREATE TABLE `subscription_plan
s` (
  `id` bigint NOT NULL AUTO_INC
REMENT,
  `title` varchar(128) COLLATE
utf8mb4_unicode_ci NOT NULL,
  `subtitle` varchar(255) COLLA
TE utf8mb4_unicode_ci DEFAULT
```

```
'',
  `price_amount` decimal(10,6)
NOT NULL DEFAULT '0.000000',
  `currency` varchar(8) COLLATE
utf8mb4_unicode_ci NOT NULL DEF
AULT 'USD',
  `duration_unit` varchar(16) C
OLLATE utf8mb4_unicode_ci NOT N
ULL DEFAULT 'month',
  `duration_value` bigint NOT N
ULL DEFAULT '1',
  `custom_seconds` bigint NOT N
ULL DEFAULT '0',
  `enabled` tinyint(1) DEFAULT
'1',
  `sort_order` bigint DEFAULT
'0',
  `stripe_price_id` varchar(12
8) COLLATE utf8mb4_unicode_ci D
EFAULT '',
  `creem_product_id` varchar(12
8) COLLATE utf8mb4_unicode_ci D
EFAULT '',
  `max_purchase_per_user` bigin
t DEFAULT '0',
  `upgrade_group` varchar(64) C
OLLATE utf8mb4_unicode_ci DEFAU
LT '',
  `total_amount` bigint NOT NUL
```

```
L DEFAULT '0',
  `quota_reset_period` varchar
(16) COLLATE utf8mb4_unicode_ci
DEFAULT 'never',
  `quota_reset_custom_seconds`
bigint DEFAULT '0',
  `created_at` bigint DEFAULT N
ULL,
  `updated_at` bigint DEFAULT N
ULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET
=utf8mb4 COLLATE=utf8mb4_unicod
e_ci;
```

订阅预消费表

```
-- ----------------------------
-- Table structure for subscrip
tion_pre_consume_records
-- ----------------------------
DROP TABLE IF EXISTS `subscript
ion_pre_consume_records`;
CREATE TABLE `subscription_pre_
consume_records` (
  `id` bigint NOT NULL AUTO_INC
REMENT,
```

```
  `request_id` varchar(64) COLL
ATE utf8mb4_unicode_ci DEFAULT
NULL,
  `user_id` bigint DEFAULT NUL
L,
  `user_subscription_id` bigint
DEFAULT NULL,
  `pre_consumed` bigint NOT NUL
L DEFAULT '0',
  `status` varchar(32) COLLATE
utf8mb4_unicode_ci DEFAULT NUL
L,
  `created_at` bigint DEFAULT N
ULL,
  `updated_at` bigint DEFAULT N
ULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `idx_subscription_
pre_consume_records_request_id`
(`request_id`),
  KEY `idx_subscription_pre_con
sume_records_status` (`status
`),
  KEY `idx_subscription_pre_con
sume_records_updated_at` (`upda
ted_at`),
  KEY `idx_subscription_pre_con
sume_records_user_id` (`user_id
`),
```

```
  KEY `idx_subscription_pre_consume_records_user_subscription_id` (`user_subscription_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

任务表（视频任务）

```
-- ----------------------------
-- Table structure for tasks
-- ----------------------------
DROP TABLE IF EXISTS `tasks`;
CREATE TABLE `tasks` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `created_at` bigint DEFAULT NULL,
  `updated_at` bigint DEFAULT NULL,
  `task_id` varchar(191) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `platform` varchar(30) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `user_id` bigint DEFAULT NUL
```

```
L,
  `group` varchar(50) COLLATE u
tf8mb4_unicode_ci DEFAULT NULL,
  `channel_id` bigint DEFAULT N
ULL,
  `quota` bigint DEFAULT NULL,
  `action` varchar(40) COLLATE
utf8mb4_unicode_ci DEFAULT NUL
L,
  `status` varchar(20) COLLATE
utf8mb4_unicode_ci DEFAULT NUL
L,
  `fail_reason` longtext COLLAT
E utf8mb4_unicode_ci,
  `submit_time` bigint DEFAULT
NULL,
  `start_time` bigint DEFAULT N
ULL,
  `finish_time` bigint DEFAULT
NULL,
  `progress` varchar(20) COLLAT
E utf8mb4_unicode_ci DEFAULT NU
LL,
  `properties` json DEFAULT NUL
L,
  `private_data` json DEFAULT N
ULL,
  `data` json DEFAULT NULL,
  PRIMARY KEY (`id`),
```

```
  KEY `idx_tasks_user_id` (`user_id`),
  KEY `idx_tasks_action` (`action`),
  KEY `idx_tasks_submit_time` (`submit_time`),
  KEY `idx_tasks_created_at` (`created_at`),
  KEY `idx_tasks_platform` (`platform`),
  KEY `idx_tasks_channel_id` (`channel_id`),
  KEY `idx_tasks_status` (`status`),
  KEY `idx_tasks_start_time` (`start_time`),
  KEY `idx_tasks_finish_time` (`finish_time`),
  KEY `idx_tasks_progress` (`progress`),
  KEY `idx_tasks_task_id` (`task_id`)
) ENGINE=InnoDB AUTO_INCREMENT=82 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

令牌表

```
-- ----------------------------
-- Table structure for tokens
-- ----------------------------
DROP TABLE IF EXISTS `tokens`;
CREATE TABLE `tokens` (
  `id` bigint NOT NULL AUTO_INC
REMENT,
  `user_id` bigint DEFAULT NUL
L,
  `key` char(48) COLLATE utf8mb
4_unicode_ci DEFAULT NULL,
  `status` bigint DEFAULT '1',
  `name` varchar(191) COLLATE u
tf8mb4_unicode_ci DEFAULT NULL,
  `created_time` bigint DEFAULT
NULL,
  `accessed_time` bigint DEFAUL
T NULL,
  `expired_time` bigint DEFAULT
'-1',
  `remain_quota` bigint DEFAULT
'0',
  `unlimited_quota` tinyint(1)
DEFAULT NULL,
  `model_limits_enabled` tinyin
t(1) DEFAULT NULL,
  `model_limits` text COLLATE u
tf8mb4_unicode_ci,
  `allow_ips` varchar(191) COLL
```

```
ATE utf8mb4_unicode_ci DEFAULT
'',
  `used_quota` bigint DEFAULT
'0',
  `group` varchar(191) COLLATE
utf8mb4_unicode_ci DEFAULT '',
  `cross_group_retry` tinyint
(1) DEFAULT NULL,
  `deleted_at` datetime(3) DEFA
ULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `idx_tokens_key`
(`key`),
  KEY `idx_tokens_user_id` (`us
er_id`),
  KEY `idx_tokens_name` (`name
`),
  KEY `idx_tokens_deleted_at`
(`deleted_at`)
) ENGINE=InnoDB AUTO_INCREMENT=
218 DEFAULT CHARSET=utf8mb4 COL
LATE=utf8mb4_unicode_ci;
```

充值记录表

```
-- ----------------------------
-- Table structure for top_ups
```

```
-- ----------------------------
DROP TABLE IF EXISTS `top_ups`;
CREATE TABLE `top_ups` (
  `id` bigint NOT NULL AUTO_INC
REMENT,
  `user_id` bigint DEFAULT NUL
L,
  `amount` bigint DEFAULT NULL,
  `money` double DEFAULT NULL,
  `trade_no` varchar(255) COLLA
TE utf8mb4_unicode_ci DEFAULT N
ULL,
  `payment_method` varchar(50)
COLLATE utf8mb4_unicode_ci DEFA
ULT NULL,
  `create_time` bigint DEFAULT
NULL,
  `complete_time` bigint DEFAUL
T NULL,
  `status` longtext COLLATE utf
8mb4_unicode_ci,
  PRIMARY KEY (`id`),
  UNIQUE KEY `trade_no` (`trade
_no`),
  KEY `idx_top_ups_user_id` (`u
ser_id`),
  KEY `idx_top_ups_trade_no` (`
trade_no`)
) ENGINE=InnoDB AUTO_INCREMENT=
```

```
37 DEFAULT CHARSET=utf8mb4 COLL
ATE=utf8mb4_unicode_ci;
```

```
-- ----------------------------
-- Table structure for two_fa_b
ackup_codes
-- ----------------------------
DROP TABLE IF EXISTS `two_fa_ba
ckup_codes`;
CREATE TABLE `two_fa_backup_cod
es` (
  `id` bigint NOT NULL AUTO_INC
REMENT,
  `user_id` bigint NOT NULL,
  `code_hash` varchar(255) COLL
ATE utf8mb4_unicode_ci NOT NUL
L,
  `is_used` tinyint(1) DEFAULT
NULL,
  `used_at` datetime(3) DEFAULT
NULL,
  `created_at` datetime(3) DEFA
ULT NULL,
  `deleted_at` datetime(3) DEFA
ULT NULL,
```

```
  PRIMARY KEY (`id`),
  KEY `idx_two_fa_backup_codes_
user_id` (`user_id`),
  KEY `idx_two_fa_backup_codes_
deleted_at` (`deleted_at`)
) ENGINE=InnoDB AUTO_INCREMENT=
5 DEFAULT CHARSET=utf8mb4 COLLA
TE=utf8mb4_unicode_ci;

-- ----------------------------
-- Table structure for two_fas
-- ----------------------------
DROP TABLE IF EXISTS `two_fas`;
CREATE TABLE `two_fas` (
  `id` bigint NOT NULL AUTO_INC
REMENT,
  `user_id` bigint NOT NULL,
  `secret` varchar(255) COLLATE
utf8mb4_unicode_ci NOT NULL,
  `is_enabled` tinyint(1) DEFAU
LT NULL,
  `failed_attempts` bigint DEFA
ULT '0',
  `locked_until` datetime(3) DE
FAULT NULL,
  `last_used_at` datetime(3) DE
FAULT NULL,
  `created_at` datetime(3) DEFA
ULT NULL,
```

```
  `updated_at` datetime(3) DEFA
ULT NULL,
  `deleted_at` datetime(3) DEFA
ULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `user_id` (`user_i
d`),
  KEY `idx_two_fas_user_id` (`u
ser_id`),
  KEY `idx_two_fas_deleted_at`
(`deleted_at`)
) ENGINE=InnoDB AUTO_INCREMENT=
2 DEFAULT CHARSET=utf8mb4 COLLA
TE=utf8mb4_unicode_ci;
```

用户OAuth绑定表

```
-- ----------------------------
-- Table structure for user_oau
th_bindings
-- ----------------------------
DROP TABLE IF EXISTS `user_oaut
h_bindings`;
CREATE TABLE `user_oauth_bindin
gs` (
  `id` bigint NOT NULL AUTO_INC
REMENT,
```

```
  `user_id` bigint NOT NULL,
  `provider_id` bigint NOT NUL
L,
  `provider_user_id` varchar(25
6) COLLATE utf8mb4_unicode_ci N
OT NULL,
  `created_at` datetime(3) DEFA
ULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `ux_user_provider`
(`user_id`,`provider_id`),
  UNIQUE KEY `ux_provider_useri
d` (`provider_id`,`provider_use
r_id`)
) ENGINE=InnoDB AUTO_INCREMENT=
63 DEFAULT CHARSET=utf8mb4 COLL
ATE=utf8mb4_unicode_ci;
```

用户订阅表

```
-- ----------------------------
-- Table structure for user_sub
scriptions
-- ----------------------------
DROP TABLE IF EXISTS `user_subs
criptions`;
CREATE TABLE `user_subscription
```

```
s` (
  `id` bigint NOT NULL AUTO_INC
REMENT,
  `user_id` bigint DEFAULT NUL
L,
  `plan_id` bigint DEFAULT NUL
L,
  `amount_total` bigint NOT NUL
L DEFAULT '0',
  `amount_used` bigint NOT NULL
DEFAULT '0',
  `start_time` bigint DEFAULT N
ULL,
  `end_time` bigint DEFAULT NUL
L,
  `status` varchar(32) COLLATE
utf8mb4_unicode_ci DEFAULT NUL
L,
  `source` varchar(32) COLLATE
utf8mb4_unicode_ci DEFAULT 'ord
er',
  `last_reset_time` bigint DEFA
ULT '0',
  `next_reset_time` bigint DEFA
ULT '0',
  `upgrade_group` varchar(64) C
OLLATE utf8mb4_unicode_ci DEFAU
LT '',
  `prev_user_group` varchar(64)
```

```
COLLATE utf8mb4_unicode_ci DEFA
ULT '',
  `created_at` bigint DEFAULT N
ULL,
  `updated_at` bigint DEFAULT N
ULL,
  PRIMARY KEY (`id`),
  KEY `idx_user_subscriptions_p
lan_id` (`plan_id`),
  KEY `idx_user_subscriptions_e
nd_time` (`end_time`),
  KEY `idx_user_subscriptions_s
tatus` (`status`),
  KEY `idx_user_subscriptions_n
ext_reset_time` (`next_reset_ti
me`),
  KEY `idx_user_subscriptions_u
ser_id` (`user_id`),
  KEY `idx_user_sub_active` (`u
ser_id`,`status`,`end_time`)
) ENGINE=InnoDB DEFAULT CHARSET
=utf8mb4 COLLATE=utf8mb4_unicod
e_ci;
```

用户表

```
-- ----------------------------
-- Table structure for users
-- ----------------------------
DROP TABLE IF EXISTS `users`;
CREATE TABLE `users` (
  `id` bigint NOT NULL AUTO_INC
REMENT,
  `username` varchar(191) COLLA
TE utf8mb4_unicode_ci DEFAULT N
ULL,
  `password` longtext COLLATE u
tf8mb4_unicode_ci NOT NULL,
  `display_name` varchar(191) C
OLLATE utf8mb4_unicode_ci DEFAU
LT NULL,
  `role` bigint DEFAULT '1',
  `status` bigint DEFAULT '1',
  `email` varchar(191) COLLATE
utf8mb4_unicode_ci DEFAULT NUL
L,
  `github_id` varchar(191) COLL
ATE utf8mb4_unicode_ci DEFAULT
NULL,
  `discord_id` varchar(191) COL
LATE utf8mb4_unicode_ci DEFAULT
NULL,
  `oidc_id` varchar(191) COLLAT
E utf8mb4_unicode_ci DEFAULT NU
LL,
```

```
  `wechat_id` varchar(191) COLL
ATE utf8mb4_unicode_ci DEFAULT
NULL,
  `telegram_id` varchar(191) CO
LLATE utf8mb4_unicode_ci DEFAUL
T NULL,
  `access_token` char(32) COLLA
TE utf8mb4_unicode_ci DEFAULT N
ULL,
  `quota` bigint DEFAULT '0',
  `used_quota` bigint DEFAULT
'0',
  `request_count` bigint DEFAUL
T '0',
  `group` varchar(64) COLLATE u
tf8mb4_unicode_ci DEFAULT 'defa
ult',
  `aff_code` varchar(32) COLLAT
E utf8mb4_unicode_ci DEFAULT NU
LL,
  `aff_count` bigint DEFAULT
'0',
  `aff_quota` bigint DEFAULT
'0',
  `aff_history` bigint DEFAULT
'0',
  `inviter_id` bigint DEFAULT N
ULL,
  `deleted_at` datetime(3) DEFA
```

```
ULT NULL,
  `linux_do_id` varchar(191) CO
LLATE utf8mb4_unicode_ci DEFAUL
T NULL,
  `setting` text COLLATE utf8mb
4_unicode_ci,
  `remark` varchar(255) COLLATE
utf8mb4_unicode_ci DEFAULT NUL
L,
  `stripe_customer` varchar(64)
COLLATE utf8mb4_unicode_ci DEFA
ULT NULL,
  `created_at` bigint DEFAULT N
ULL,
  `phone` varchar(20) COLLATE u
tf8mb4_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`usern
ame`),
  UNIQUE KEY `idx_users_access_
token` (`access_token`),
  UNIQUE KEY `idx_users_aff_cod
e` (`aff_code`),
  KEY `idx_users_discord_id` (`
discord_id`),
  KEY `idx_users_oidc_id` (`oid
c_id`),
  KEY `idx_users_we_chat_id` (`
wechat_id`),
```

```
  KEY `idx_users_telegram_id` (`telegram_id`),
  KEY `idx_users_display_name` (`display_name`),
  KEY `idx_users_inviter_id` (`inviter_id`),
  KEY `idx_users_deleted_at` (`deleted_at`),
  KEY `idx_users_linux_do_id` (`linux_do_id`),
  KEY `idx_users_stripe_customer` (`stripe_customer`),
  KEY `idx_users_username` (`username`),
  KEY `idx_users_email` (`email`),
  KEY `idx_users_git_hub_id` (`github_id`),
  KEY `idx_users_phone` (`phone`)
) ENGINE=InnoDB AUTO_INCREMENT=888888967 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

提供商表

```
-- ----------------------------
-- Table structure for vendors
-- ----------------------------
DROP TABLE IF EXISTS `vendors`;
CREATE TABLE `vendors` (
  `id` bigint NOT NULL AUTO_INC
REMENT,
  `name` varchar(128) COLLATE u
tf8mb4_unicode_ci NOT NULL,
  `description` text COLLATE ut
f8mb4_unicode_ci,
  `icon` varchar(128) COLLATE u
tf8mb4_unicode_ci DEFAULT NULL,
  `status` bigint DEFAULT '1',
  `created_time` bigint DEFAULT
NULL,
  `updated_time` bigint DEFAULT
NULL,
  `deleted_at` datetime(3) DEFA
ULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `uk_vendor_name_de
lete_at` (`name`,`deleted_at`),
  KEY `idx_vendors_deleted_at`
(`deleted_at`)
) ENGINE=InnoDB AUTO_INCREMENT=
33 DEFAULT CHARSET=utf8mb4 COLL
ATE=utf8mb4_unicode_ci;
```

```
SET FOREIGN_KEY_CHECKS = 1;
```

Token支付流程安全问题

安全流程SQL

```
## GO语法
Update("quota", gorm.Expr("quota + ?", quota))
==>
## 转换就SQL
UPDATE user SET quota = quota + ? WHERE id = ?

Update("quota", gorm.Expr("quota - ?", quota))
==>
UPDATE user SET quota = quota - ? WHERE id = ?
```

说明

在 MySQL / PostgreSQL 中是：

✔ 原子操作（Atomic）
不会出现“先读再写”的竞态问题
多个请求同时执行时，数据库会串行处理行锁
✔ 行级锁（InnoDB）
同一行（id）会被锁住
并发执行时不会丢失更新

所以不会出现这种问题：

```
线程A：读 quota=100
线程B：读 quota=100
线程A：写 110
线程B：写 110 （丢失更新）
```

风险点

1. quota 可能被扣成负数

支付的核心流程

支付核心流程：

1. 支付请求 (RequestBfPay): 计算金额(元→分) → 创建本地订单 → RSA2签名 → 调用慧收钱 NativePay API → 返回二维码链接 (qr_code)
2. 支付回调 (BfWebhook): 接收 POST 回调 → RSA2验签 → 解析业务数据 → SUCCESS时调用 RechargeBf 充值 → 返回 SUCCESS 字符串

接口说明

接口编写文件 `controller/topup_bf.go`

- 新增 `RequestBfPay` 支付回调接口
- 新增 `BfWebhook` 处理回调后的内容

修改文件

fai-hub-backend/model/topup.go

```
可选
- 查询订单 接口

参考API
```

https://docs.baofu.com/docs/interface_document/interface_document-Native-TransNotify

```
 ## 支付请求

```mermaid
sequenceDiagram
    participant Client as 客户端
    participant relayRouter as Relay路由
    participant Server as 业务服务
    participant BfPay as 慧收钱

    Client->>relayRouter: 发起支付请求
```
```

```
    relayRouter->>Server: 转发请求

    Server->>Server: 金额转换（元→分）
    Server->>Server: 创建本地订单
    Server->>Server: RSA2签名

    Server->>BfPay: 调用 Native Pay API
    BfPay-->>Server: 返回 qr_code

    Server-->>relayRouter: 返回二维码链接
    relayRouter-->>Client: 返回二维码

    Client->>Client: 展示二维码扫码支付
```

支付回调

需要配置的选项

- BfEnabled = true
- BfMerchantNo = 商户号
- BfPrivateKey = 商户RSA私钥(PEM)
- BfPublicKey = 平台RSA公钥

  (PEM，用于验证回调签名)

- BfApiUrl = API地址（默认 https://api.huishouqian.com/api/acquiring，测试环境用 https://test-api.huishouqian.com/api/acquiring）
- BfUnitPrice = 单价（可选，默认 1.0）
- BfMinTopUp = 最小充值（可选，默认1）
- BfPayType = 支付类型（可选，默认 DYNAMIC_ALL）

根据模型阶梯扣费SQL

```
-- 插入消费记录
INSERT INTO `logs` (
    user_id,
    created_at,
    type,
    content,
    username,
    token_name,
    model_name,
    quota,
    prompt_tokens,
    completion_tokens,
    use_time,
    is_stream,
    channel_id,
    channel_name,
    token_id,
    `group`,
    ip,
    request_id,
    other,
    org_id
)
SELECT
user_id,
created_at,
```

```
`type`,
content,
username,
token_name,
model_name,
quota,
prompt_tokens,
completion_tokens,
use_time,
is_stream,
channel_id,
channel_name,
token_id,
`group`,
ip,
request_id,
other,
org_id
FROM
    `logs` t
WHERE
    user_id = 888888961
    AND t.model_name = 'gpt-5.5'
    AND t.prompt_tokens > 272000
    AND created_at >= UNIX_TIMESTAMP(
    CURDATE())
    AND created_at < UNIX_TIMESTAMP(
    DATE_ADD( CURDATE(), INTERVAL 1 DAY ));
```

```sql
-- 查询 使用tokensshu
SELECT
user_id,
sum(prompt_tokens) ,
sum(completion_tokens),
sum(JSON_EXTRACT( other, '$.cache_write_tokens' )) AS `cache_write_tokens`,
sum(JSON_EXTRACT( other, '$.cache_tokens' )) AS `cache_tokens`
FROM
    `logs` t
WHERE
    user_id = 888888961
    AND t.model_name = 'gpt-5.5'
    AND t.prompt_tokens > 272000
    AND created_at >= UNIX_TIMESTAMP(
    CURDATE())
    AND created_at < UNIX_TIMESTAMP(
    DATE_ADD( CURDATE(), INTERVAL 1 DAY ));
    GROUP BY user_id
```

统计-模型错误占比

按日期分类-查询模型正确率

```
SELECT
  stat_date AS 日期,
  total_all AS 总请求数,

  `成功-1次请求` AS 一次成功数量,
  CONCAT(ROUND(`成功-1次请求` / total_all * 100,2),'%') AS 一次成功占比,

  `成功-2次请求` AS 二次成功数量,
  CONCAT(ROUND(`成功-2次请求` / total_all * 100,2),'%') AS 二次成功占比,

  `成功-3次请求` AS 三次成功数量,
  CONCAT(ROUND(`成功-3次请求` / total_all * 100,2),'%') AS 三次
```

```
成功占比,

  `失败-1次请求` AS 一次失败数量,
  CONCAT(ROUND(`失败-1次请求` / total_all * 100,2),'%') AS 一次失败占比,

  `失败-2次请求` AS 二次失败数量,
  CONCAT(ROUND(`失败-2次请求` / total_all * 100,2),'%') AS 二次失败占比,

  `失败-3次请求` AS 三次失败数量,
  CONCAT(ROUND(`失败-3次请求` / total_all * 100,2),'%') AS 三次失败占比,

  other AS 其他类型数量,
  CONCAT(ROUND(other / total_all * 100,2),'%') AS 其他类型占比,

  success_rate AS 整体成功率,
  fail_rate AS 整体失败率

FROM (
```

```
  SELECT

    stat_date,

    COUNT(DISTINCT request_id)
AS total_all,

    SUM(IF(title = '成功-1次请
求',1,0)) AS `成功-1次请求`,
    SUM(IF(title = '成功-2次请
求',1,0)) AS `成功-2次请求`,
    SUM(IF(title = '成功-3次请
求',1,0)) AS `成功-3次请求`,

    SUM(IF(title = '失败-1次请
求',1,0)) AS `失败-1次请求`,
    SUM(IF(title = '失败-2次请
求',1,0)) AS `失败-2次请求`,
    SUM(IF(title = '失败-3次请
求',1,0)) AS `失败-3次请求`,

    SUM(IF(title = '其他类型',
1,0)) AS other,

    CONCAT(
      ROUND(
        SUM(IF(title LIKE '成
功%',1,0))
        /
```

```
        COUNT(DISTINCT request_
id)
        *100
      ,2),
      '%'
    ) AS success_rate,

    CONCAT(
      ROUND(
        SUM(IF(title LIKE '失
败%',1,0))
        /
        COUNT(DISTINCT request_
id)
        *100
      ,2),
      '%'
    ) AS fail_rate


  FROM (

    SELECT
      DATE(FROM_UNIXTIME(create
d_at)) AS stat_date,

      request_id,

      SUM(type) AS type_times,
```

```
      CASE SUM(type)
        WHEN 2 THEN '成功-1次请
求'
        WHEN 7 THEN '成功-2次请
求'
        WHEN 12 THEN '成功-3次
请求'

        WHEN 5 THEN '失败-1次请
求'
        WHEN 10 THEN '失败-2次
请求'
        WHEN 15 THEN '失败-3次
请求'

        ELSE '其他类型'
      END AS title


    FROM logs

    WHERE
      user_id = '888889422'
      AND type IN (2,5)
      AND model_name = 'claude-
fable-5'

    GROUP BY
```

```
      DATE(FROM_UNIXTIME(create
d_at)),
      request_id

  ) t1

  GROUP BY stat_date

) t2

ORDER BY stat_date DESC;
```

按小时分类

```
SELECT
  stat_hour AS 小时,
  total_all AS 总请求数,

  `成功-1次请求` AS 一次成功数
量,
  CONCAT(ROUND(`成功-1次请求` /
total_all * 100,2),'%') AS 一次
成功占比,

  `成功-2次请求` AS 二次成功数
量,
  CONCAT(ROUND(`成功-2次请求` /
total_all * 100,2),'%') AS 二次
```

```
成功占比,

  `成功-3次请求` AS 三次成功数量,
  CONCAT(ROUND(`成功-3次请求` / total_all * 100,2),'%') AS 三次成功占比,

  `失败-1次请求` AS 一次失败数量,
  CONCAT(ROUND(`失败-1次请求` / total_all * 100,2),'%') AS 一次失败占比,

  `失败-2次请求` AS 二次失败数量,
  CONCAT(ROUND(`失败-2次请求` / total_all * 100,2),'%') AS 二次失败占比,

  `失败-3次请求` AS 三次失败数量,
  CONCAT(ROUND(`失败-3次请求` / total_all * 100,2),'%') AS 三次失败占比,

  other AS 其他类型数量,
  CONCAT(ROUND(other / total_all * 100,2),'%') AS 其他类型占
```

```
比,

  success_rate AS 整体成功率,
  fail_rate AS 整体失败率


FROM (

  SELECT

    stat_hour,

    COUNT(DISTINCT request_id)
AS total_all,


    SUM(IF(title='成功-1次请
求',1,0)) AS `成功-1次请求`,
    SUM(IF(title='成功-2次请
求',1,0)) AS `成功-2次请求`,
    SUM(IF(title='成功-3次请
求',1,0)) AS `成功-3次请求`,


    SUM(IF(title='失败-1次请
求',1,0)) AS `失败-1次请求`,
    SUM(IF(title='失败-2次请
求',1,0)) AS `失败-2次请求`,
    SUM(IF(title='失败-3次请
```

```
求',1,0)) AS `失败-3次请求`,


    SUM(IF(title='其他类型',1,
0)) AS other,


    CONCAT(
      ROUND(
        SUM(IF(title LIKE '成
功%',1,0))
        /
        COUNT(DISTINCT request_
id)
        *100,
      2),
      '%'
    ) AS success_rate,


    CONCAT(
      ROUND(
        SUM(IF(title LIKE '失
败%',1,0))
        /
        COUNT(DISTINCT request_
id)
        *100,
      2),
```

```
      '%'
    ) AS fail_rate



  FROM (

    SELECT

      DATE_FORMAT(
        FROM_UNIXTIME(created_a
t),
        '%Y-%m-%d %H:00:00'
      ) AS stat_hour,


      request_id,


      SUM(type) AS type_times,


      CASE SUM(type)

        WHEN 2 THEN '成功-1次请
求'
        WHEN 7 THEN '成功-2次请
求'
        WHEN 12 THEN '成功-3次
```

```
请求'

        WHEN 5 THEN '失败-1次请
求'
        WHEN 10 THEN '失败-2次
请求'
        WHEN 15 THEN '失败-3次
请求'

        ELSE '其他类型'

      END AS title


    FROM logs


    WHERE
      user_id='888889422'
      AND type IN (2,5)
      AND model_name='claude-fa
ble-5'


    GROUP BY

      DATE_FORMAT(
        FROM_UNIXTIME(created_a
t),
```

```
        '%Y-%m-%d %H:00:00'
      ),

      request_id


  ) t1


  GROUP BY stat_hour


) t2


ORDER BY stat_hour DESC;
```

提供商超过100秒的

```
SELECT
    '提供商K' as '提供商',
    -- 总条数
    COUNT(*) AS '总笔数',
    -- use_time>100 的条数
    SUM(IF(use_time > 100, 1,
0)) AS '使用超过100秒笔数',
    -- 计算占比，保留2位小数
```

```
    CONCAT(ROUND(SUM(IF(use_tim
e > 100, 1, 0)) / COUNT(*) * 10
0, 2), '%') AS '占比'
FROM `logs`
WHERE
    channel_id = '69'
    AND created_at >= UNIX_TIME
STAMP(CURDATE()) ;
```

多提供商分组查

```
SELECT
    channel_id AS 渠道ID,
    CASE channel_id
        WHEN 69 THEN '提供商K'
        WHEN 64 THEN '提供商I'
-- 自行替换64对应的名称
    END AS 提供商,
    COUNT(*) AS 总笔数,
    SUM(IF(use_time > 100, 1,
0)) AS 使用超过100秒笔数,
    CONCAT(ROUND(SUM(IF(use_tim
e > 100, 1, 0)) / COUNT(*) * 10
0, 2), '%') AS 占比
FROM `logs`
WHERE
    channel_id IN (69,64)
    -- created_at是毫秒bigint,
```

```
必须 *1000
    AND created_at >= UNIX_TIMESTAMP(CURDATE())
GROUP BY channel_id;


SELECT
    channel_id AS 渠道ID,
        c.`name` AS 渠道名称,
    COUNT(*) AS 总笔数,
    SUM(IF(use_time > 50, 1, 0)) AS 使用超过50秒笔数,
    CONCAT(ROUND(SUM(IF(use_time > 50, 1, 0)) / COUNT(*) * 100, 2), '%') AS 超50秒占比,
    SUM(IF(use_time > 100, 1, 0)) AS 使用超过100秒笔数,
    CONCAT(ROUND(SUM(IF(use_time > 100, 1, 0)) / COUNT(*) * 100, 2), '%') AS 超100秒占比
FROM `logs` l
LEFT JOIN channels c
ON l.channel_id = c.id
WHERE
    l.channel_id IN (69,64)
    AND created_at >= UNIX_TIMESTAMP(CURDATE())
GROUP BY channel_id;
```

```sql
-- 各渠道明细数据
SELECT
    l.channel_id AS 渠道ID,
    c.`name` AS 渠道名称,
    COUNT(*) AS 总笔数,
    SUM(IF(use_time > 50, 1,
0)) AS 使用超过50秒笔数,
    CONCAT(ROUND(SUM(IF(use_tim
e > 50, 1, 0)) / COUNT(*) * 10
0, 2), '%') AS 超50秒占比,
    SUM(IF(use_time > 100, 1,
0)) AS 使用超过100秒笔数,
    CONCAT(ROUND(SUM(IF(use_tim
e > 100, 1, 0)) / COUNT(*) * 10
0, 2), '%') AS 超100秒占比
FROM `logs` l
LEFT JOIN channels c ON l.chann
el_id = c.id
WHERE created_at >= UNIX_TIMEST
AMP(CURDATE())
GROUP BY l.channel_id, c.`name`

UNION ALL

-- 全局汇总行
SELECT
    NULL AS 渠道ID,
    '汇总合计' AS 渠道名称,
```

```
    COUNT(*) AS 总笔数,
    SUM(IF(use_time > 50, 1,
0)) AS 使用超过50秒笔数,
    CONCAT(ROUND(SUM(IF(use_tim
e > 50, 1, 0)) / COUNT(*) * 10
0, 2), '%') AS 超50秒占比,
    SUM(IF(use_time > 100, 1,
0)) AS 使用超过100秒笔数,
    CONCAT(ROUND(SUM(IF(use_tim
e > 100, 1, 0)) / COUNT(*) * 10
0, 2), '%') AS 超100秒占比
FROM `logs` l
WHERE created_at >= UNIX_TIMEST
AMP(CURDATE()) ;
```