Apusic 金蝶天燕

生成时间：2026-07-13

开发手册

金蝶Apusic负载均衡器

版权声明

本文档所涉及的软件著作权、版权等知识产权已依法进行了注册，由金蝶天燕云计算股份有限公司合法拥有。受《中华人民共和国著作权法》《计算机软件保护条例》《知识产权保护条例》和相关国际版权条约、法律、法规以及其它知识产权法律和条约的保护。未经授权许可，不得非法使用。

免责声明

本文档包含的版权信息由金蝶天燕云计算股份有限公司合法拥有，受法律的保护，金蝶天燕云计算股份有限公司对本文档可能涉及到的非金蝶天燕云计算股份有限公司的信息不承担任何责任。在法律允许的范围内，您可以查阅并仅能够在《中华人民共和国著作权法》规定的合法范围内复制和打印本文档。任何单位和个人未经金蝶天燕云计算股份有限公司书面授权许可，不得使用、修改、再发布本文档的任何部分和内容，否则将被视为侵权，金蝶天燕云计算股份有限公司有依法追究其责任的权利。

本文档如有更新，不另行通知。对本文档中的问题您可向金蝶天燕云计算股份有限公司告知或查询。未经本公司明确授予的任何权利均予保留。

商标声明

Apusic是深圳市金蝶天燕云计算股份有限公司向中华人民共和国国家商标局申请注册的注册商标，注册商标专用权由金蝶天燕合法拥有，受法律保护。未经金蝶天燕的书面许可，任何单位及个人不得以任何方式或理由对该商标的任何部分进行使用、复制、修改、传播、抄录或与其它产品捆绑使用销售。凡侵犯金蝶天燕商标权的，金蝶天燕将依法追究其法律责任。本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

目录

1 前言

本文档为金蝶Apusic负载均衡器软件V2.0.6标准版的开发手册，面向运维与开发人员，详细介绍 ALB 管控台对外提供的**外部 API 接口**、**alb-cli 命令行工具** 以及面向 **AI Agent（Claude、OpenCode 等）** 的集成能力，帮助客户基于 ALB 实现自动化运维、CI/CD 发布、平台对接和 AI 辅助运维等场景。

1.1 适用对象

本文档适用于：

- ALB 二次开发与集成工程师
- 运维自动化平台 / DevOps 平台开发工程师
- 引入 AI Agent 进行日常运维的 SRE 工程师
- ERP、运维监控、业务发布等业务系统的对接方
- 客户单位的系统架构师、技术负责人

1.2 相关文档

了解更多ALB V2.0.6产品相关的信息，请参阅以下ALB V2.0.6产品手册文档集：

序号	手册文档	说明
1	金蝶Apusic负载均衡器软件 V2.0.6 发版说明	介绍了ALB 产品版本更新内容 。
2	金蝶Apusic负载均衡器软件 V2.0.6 快速使用手册	简单介绍了如何快速上手使用ALB 。
3	金蝶Apusic负载均衡器软件 V2.0.6 安装手册	详细介绍如何在各操作系统上安装ALB，以及ALB服务启停操作，产品的注册过程。
4	金蝶Apusic负载均衡器软件 V2.0.6 代理功能手册	详细介绍 ALB 相关功能的使用、配置、管理及配套工具的使用方法。
5	金蝶Apusic负载均衡器软件 V2.0.6 管控台用户手册	详细介绍ALB管控台相关功能的使用和操作说明。
6	金蝶Apusic负载均衡器软件 V2.0.6 开发手册	详细介绍ALB外部API、alb-cli命令行工具及AI Agent集成方法。
7	金蝶Apusic负载均衡器软件 V2.0.6 迁移手册	详细介绍ALB历史版本迁移升级到V2.0.6版本的说明。

8	金蝶Apusic负载均衡器软件 V2.0.6 运维手册	详细介绍ALB的监控、运维、安全加固等运维说明。
9	金蝶Apusic负载均衡器软件 V2.0.6 性能优化手册	详细介绍ALB性能调优的说明。

1.3 技术支持

ALB产品提供全面的技术支持服务，您可以通过以下方式获得技术支持：

- 网址：www.apusic.com
- 电话：400-855-5800
- 邮箱：support@apusic.com
- 金蝶云社区：https://vip.kingdee.com/?productId=73&productLineId=14&lang=zh-CN

您在取得技术支持时，请提供如下信息：

1. 您的姓名
2. 公司信息与联系方式
3. 操作系统及其版本
4. 产品版本号
5. 出现异常及错误的日志、截图等详细信息

2 目录

---

3 概述

ALB V2.0.6 标准版在管控台基础上，开放了三种面向开发与运维人员的能力：

能力	形态	适用场景
外部 Site API	HTTPS RESTful API（API Key 鉴权）	业务系统、运维平台、CI/CD 流水线对接
alb-cli	单二进制命令行工具	运维脚本、自动化任务、人工快速操作
AI Agent SKILL	面向 Claude、OpenCode 等 Agent 的 SKILL 描述	引入 AI 助手进行日常运维和故障排查

3.1 三种能力对比

维度	外部 API	alb-cli	AI Agent SKILL
调用方式	HTTP/JSON	命令行	自然语言 + 工具调用
鉴权	`X-API-Key`	`ALB_API_KEY` 环境变量	`ALB_API_KEY` 环境变量
适合集成方	后端服务、ERP	运维、DevOps	AI 助手场景
输出格式	JSON	JSON / Table	解析 JSON 后给出回答
危险操作	需自行加确认	内置交互确认	内置用户确认 + 非 TTY 拒绝

> ***注意***：*以上三种能力底层共用同一套管控台接口，调用语义保持一致。开发人员可根据集成场景选择最合适的形态。*

3.2 读者路径

- **后端开发者**：先阅读 外部 API 接口，重点关注通用约定、HTTP 站点、Stream 站点、HA 集群行为。
- **运维 / DevOps**：先阅读 alb-cli 命令行工具 和 典型集成场景。
- **AI 集成方**：先阅读 AI Agent 集成 和 典型集成场景 中的故障排查流程。

---

4 外部 API 接口

> *版本：* *v1.0*
> *基础路径：* `/api/alb/console/external/v1`
> *适用版本：* *ALB V2.0.6+*

4.1 通用约定

4.1.1 鉴权方式

所有外部 API 接口通过 HTTP Header 传递 API Key：

```
X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

API Key 在 `【ALB安装目录】/console/conf/config.yaml` 中配置：

```
external:
  enabled: true
  api_keys:
    - name: "erp-system"
      key: "alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

> ***安全提示***：*请妥善保管 API Key，生产环境请通过 HTTPS 访问管控台；建议为不同业务系统分配不同的 API Key，便于审计与按需吊销。*

4.1.2 请求格式

- 请求体统一使用 `application/json`
- 路径参数 `:id` 为站点 ID（数字）
- `locations`、`upstreams`、`backend_servers` 字段为 JSON **字符串**（后端按字符串存储），请求时需将数组序列化为 JSON 字符串，如 `"[{\"path\":\"/\",\"type\":\"proxy\"}]"`

4.1.3 响应格式

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

```
  "msg": "ok"
}
```

code	含义
0	成功
7	通用业务错误
401	鉴权失败（缺少 API Key / Key 无效）
403	外部 API 未启用
10001	当前节点为 HA 备节点，禁止写入

4.1.4 HA 集群行为

节点状态	读操作 (GET)	写操作 (POST/PUT/DELETE)
未启用 HA	✅ 正常	✅ 正常
启用 HA + 主节点	✅ 正常	✅ 正常 + **自动同步到备节点**
启用 HA + 备节点	✅ 正常	❌ 返回 code=10001

> ***提示***：*HA 备节点的读操作不会自动转发到主节点，如需从备节点获取最新数据，请将请求发往主节点，或在客户端做重试。*

4.2 HTTP 站点接口

4.2.1 创建 HTTP 站点

POST `/sites/http`

请求参数

字段	类型	必填	说明
name	string	是	站点名称
domains	string	是	监听域名
listen_port	int	是	监听端口
locations	string(JSON)	否	location 配置列表（JSON 字符串）

upstreams	string(JSON)	否	upstream 配置列表（JSON 字符串），locations 中使用 upstream_name 时需传入
ssl_cert_id	int	否	SSL 证书 ID
auto_reload	bool	否	保存后自动重载，默认 true
status	string	否	`enabled` / `disabled`，默认 `enabled`

Location 字段说明

字段	类型	必填	说明
path	string	是	匹配路径，如 `/` 或 `/api`
type	string	是	`proxy` / `static` / `redirect`
proxy_pass	string	否	直接填写后端地址，如 `http://192.168.1.10:8080`
upstream_name	string	否	引用已有的 upstream 名称（需同时在 `upstreams` 中定义）

> ***注意**：`proxy_pass` 和 `upstream_name` 二选一。使用 `upstream_name` 时必须同时在 `upstreams` 字段中定义该 upstream，否则 nginx 重载会失败。*

请求示例 1：直接代理（最简单，无需 upstream）

```
curl -X POST "http://alb-
master:8887/api/alb/console/external/v1/sites/http" \
  -H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "test-api",
    "domains": "www.example.com",
    "listen_port": 8088,
    "locations": "
[{\"path\":\"/\",\"type\":\"proxy\",\"proxy_pass\":\"http://192.168.1.1
[]}]",
    "upstreams": "[]",
    "custom_headers": "[]",
    "proxy_set_headers": "[]",
    "extra_directives": "[]",
```

```
    "auto_reload": true,
    "status": "enabled"
  }'
```

请求示例 2：使用 Upstream（支持负载均衡）

```
curl -X POST "http://alb-
master:8887/api/alb/console/external/v1/sites/http" \
  -H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "test-upstream",
    "domains": "www.example.com",
    "listen_port": 8090,
    "upstreams": "[{\"name\":\"backend-
web\",\"balance_method\":\"round-robin\",\"servers\":
[{\"address\":\"192.168.1.10:8080\",\"weight\":1},
{\"address\":\"192.168.1.11:8080\",\"weight\":1}]}]",
    "locations": "
[{\"path\":\"/\",\"type\":\"proxy\",\"upstream_name\":\"backend-
web\",\"proxy_timeout\":60,\"support_web_socket\":false,\"use_alias\":f
[]}]",
    "custom_headers": "[]",
    "proxy_set_headers": "[]",
    "extra_directives": "[]",
    "auto_reload": true,
    "status": "enabled"
  }'
```

响应示例

```
{
  "code": 0,
  "data": {
    "id": 42,
```

```
    "name": "test-upstream",
    "domains": "www.example.com",
    "listen_port": 8090,
    "status": "enabled"
  },
  "msg": "创建成功"
}
```

4.2.2 更新 HTTP 站点

PUT `/sites/http`

请求参数

字段	类型	必填	说明
id	int	是	站点 ID
name	string	否	站点名称
domains	string	否	监听域名
listen_port	int	否	监听端口
locations	string(JSON)	否	location 配置列表（JSON 字符串）
upstreams	string(JSON)	否	upstream 配置列表（JSON 字符串）
ssl_cert_id	int	否	SSL 证书 ID
auto_reload	bool	否	保存后自动重载
status	string	否	`enabled` / `disabled`

请求示例

```
curl -X PUT "http://alb-
master:8887/api/alb/console/external/v1/sites/http" \
  -H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
  -H "Content-Type: application/json" \
  -d '{
    "id": 42,
    "name": "test-api-update",
```

```
    "listen_port": 8080
  }'
```

响应示例

```
{
  "code": 0,
  "data": null,
  "msg": "更新成功"
}
```

4.2.3 删除 HTTP 站点

DELETE `/sites/http`

请求参数

字段	类型	必填	说明
id	int	是	站点 ID

请求示例

```
curl -X DELETE "http://alb-
master:8887/api/alb/console/external/v1/sites/http" \
  -H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
  -H "Content-Type: application/json" \
  -d '{
    "id": 42
  }'
```

响应示例

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

```
  "msg": "删除成功"
}
```

4.2.4 启用 HTTP 站点

POST `/sites/http/{id}/enable`

```
curl -X POST "http://alb-
master:8887/api/alb/console/external/v1/sites/http/42/enable" \
  -H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

响应示例

```
{
  "code": 0,
  "data": null,
  "msg": "启用成功"
}
```

4.2.5 禁用 HTTP 站点

POST `/sites/http/{id}/disable`

```
curl -X POST "http://alb-
master:8887/api/alb/console/external/v1/sites/http/42/disable" \
  -H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

响应示例

```
{
  "code": 0,
  "data": null,
  "msg": "禁用成功"
}
```

4.2.6 获取 HTTP 站点列表

GET `/sites/http`

查询参数

字段	类型	必填	说明
page	int	否	页码，默认 1
pageSize	int	否	每页条数，默认 10
name	string	否	按名称模糊搜索

请求示例

```
curl "http://alb-master:8887/api/alb/console/external/v1/sites/http?page=1&pageSize=10" \
  -H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

响应示例

```
{
  "code": 0,
  "data": [
    {
      "id": 42,
      "name": "官网",
      "domains": "www.example.com",
      "listen_port": 80,
      "status": "enabled",
      "created_at": "2024-01-15 10:30:00"
    }
  ],
  "msg": "ok"
}
```

4.2.7 获取 HTTP 站点详情

GET `/sites/http/{id}`

```
curl "http://alb-
master:8887/api/alb/console/external/v1/sites/http/42" \
  -H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

响应示例

```
{
  "code": 0,
  "data": {
    "id": 42,
    "name": "官网",
    "domains": "www.example.com",
    "listen_port": 80,
    "locations": [
      {
        "path": "/",
        "type": "proxy",
        "upstream_name": "backend-web"
      }
    ],
    "status": "enabled",
    "created_at": "2024-01-15 10:30:00"
  },
  "msg": "ok"
}
```

4.2.8 为 HTTP 站点绑定 SSL 证书

POST `/sites/http/bind-ssl`

请求参数

字段	类型	必填	说明
site_id	int	是	站点 ID

cert_id	int	是	证书 ID

```
curl -X POST "http://alb-
master:8887/api/alb/console/external/v1/sites/http/bind-ssl" \
  -H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
  -H "Content-Type: application/json" \
  -d '{
    "site_id": 42,
    "cert_id": 2
  }'
```

4.3 Stream 站点接口

Stream 站点用于四层负载均衡（TCP/UDP），接口结构与 HTTP 站点一致。

4.3.1 创建 Stream 站点

POST `/sites/stream`

请求参数

字段	类型	必填	说明
name	string	是	站点名称
listen_port	int	是	监听端口
protocol	string	是	TCP / UDP
backend_servers	string(JSON)	是	后端服务器列表（JSON 字符串）
auto_reload	bool	否	保存后自动重载，默认 true
status	string	否	`enabled` / `disabled`，默认 `enabled`

BackendServer 字段说明

字段	类型	必填	说明
address	string	是	后端地址，如 `192.168.1.10:3306`
weight	int	否	权重，默认 1

max_fails	int	否	最大失败次数
fail_timeout	int	否	失败超时（秒）
backup	bool	否	是否为备份服务器

请求示例

```
curl -X POST "http://alb-
master:8887/api/alb/console/external/v1/sites/stream" \
  -H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "数据库代理",
    "listen_port": 3306,
    "protocol": "TCP",
    "backend_servers": "
[{\"address\":\"192.168.1.10:3306\",\"weight\":1}]",
    "auto_reload": true,
    "status": "enabled"
  }'
```

响应示例

```
{
  "code": 0,
  "data": {
    "id": 15,
    "name": "数据库代理",
    "listen_port": 3306,
    "protocol": "TCP",
    "status": "enabled"
  },
  "msg": "创建成功"
}
```

4.3.2 更新 Stream 站点

PUT `/sites/stream`

```
curl -X PUT "http://alb-
master:8887/api/alb/console/external/v1/sites/stream" \
  -H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
  -H "Content-Type: application/json" \
  -d '{
    "id": 15,
    "backend_servers": "
[{\"address\":\"192.168.1.11:3306\",\"weight\":1}]"
  }'
```

4.3.3 删除 Stream 站点

DELETE `/sites/stream`

```
curl -X DELETE "http://alb-
master:8887/api/alb/console/external/v1/sites/stream" \
  -H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" \
  -H "Content-Type: application/json" \
  -d '{"id": 15}'
```

4.3.4 启用 / 禁用 Stream 站点

POST `/sites/stream/{id}/enable`

POST `/sites/stream/{id}/disable`

```
curl -X POST "http://alb-
master:8887/api/alb/console/external/v1/sites/stream/15/enable" \
  -H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

curl -X POST "http://alb-
```

```
master:8887/api/alb/console/external/v1/sites/stream/15/disable" \
  -H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

4.3.5 获取 Stream 站点列表

GET `/sites/stream?page=1&pageSize=10`

```
curl "http://alb-
master:8887/api/alb/console/external/v1/sites/stream?
page=1&pageSize=10" \
  -H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

4.3.6 获取 Stream 站点详情

GET `/sites/stream/{id}`

```
curl "http://alb-
master:8887/api/alb/console/external/v1/sites/stream/15" \
  -H "X-API-Key: alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
```

4.4 错误响应示例

4.4.1 鉴权失败（401）

```
{
  "code": 401,
  "msg": "invalid api key"
}
```

4.4.2 外部 API 未启用（403）

```
{
  "code": 403,
```

```
  "msg": "外部 API 未启用"
}
```

4.4.3 HA 备节点写入（10001）

```
{
  "code": 10001,
  "msg": "当前节点为 HA 备节点，请前往主节点执行"
}
```

4.4.4 业务错误（7）

```
{
  "code": 7,
  "msg": "站点名称已存在"
}
```

4.5 快速验证脚本

```
#!/bin/bash

API_KEY="alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
BASE_URL="http://alb-master:8887/api/alb/console/external/v1"

echo "=== 1. 创建 HTTP 站点（直接代理）==="
curl -s -X POST "$BASE_URL/sites/http" \
  -H "X-API-Key: $API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "测试站点",
    "domains": "test.example.com",
    "listen_port": 80,
    "locations": "
```

```
[{\"path\":\"/\",\"type\":\"proxy\",\"proxy_pass\":\"http://192.168.1.1
[]}]",
    "upstreams": "[]",
    "custom_headers": "[]",
    "proxy_set_headers": "[]",
    "extra_directives": "[]",
    "auto_reload": true,
    "status": "enabled"
  }' | jq .

echo "=== 2. 获取列表 ==="
curl -s "$BASE_URL/sites/http?page=1&pageSize=5" \
  -H "X-API-Key: $API_KEY" | jq .

echo "=== 3. 禁用站点 ==="
curl -s -X POST "$BASE_URL/sites/http/1/disable" \
  -H "X-API-Key: $API_KEY" | jq .

echo "=== 4. 删除站点 ==="
curl -s -X DELETE "$BASE_URL/sites/http" \
  -H "X-API-Key: $API_KEY" \
  -H "Content-Type: application/json" \
  -d '{"id": 1}' | jq .
```

4.6 审计查询

外部 API 调用记录在 `sys_operation_logs` 表中，可通过以下 SQL 查询：

```
SELECT
  created_at,
  operator_name,
  oper_type,
  request_method,
  request_path,
  status
```

```
FROM sys_operation_logs
WHERE operator_name LIKE 'external:%'
ORDER BY id DESC
LIMIT 20;
```

示例输出：

created_at	operator_name	oper_type	request_method	request_path	s
2024-01-15 10:30:01	external:erp-system	外部API:创建HTTP站点	POST	/api/alb/console/external/v1/sites/http	1

5 alb-cli 命令行工具

> *版本：* *V2.0.6*
> *定位：* *面向 Claude、OpenCode 等 AI Agent 的 ALB 管控台命令行入口*

5.1 安装与配置

> *ALB V2.0.6 产品包自带* `alb-cli` *可执行程序，在* `【ALB安装路径】/utils/alb-cli` *中。*

5.1.1 鉴权方式

`alb-cli` 通过 API Key 与管控台通信，请求头为 `X-API-Key`。

配置 API Key：在后端配置文件 `【ALB安装目录】/console/conf/config.yaml` 中配置 `external.api_keys`，或通过环境变量 `ALB_API_KEY` 设置。

5.1.2 配置来源（优先级从高到低）

1. 环境变量
2. 配置文件 `~/.alb/config.yaml`

环境变量

```
export ALB_CONSOLE_URL=https://172.21.61.46:8887
export ALB_API_KEY=your-api-key
```

配置文件

```
# ~/.alb/config.yaml
url: https://172.21.61.46:8887
api_key: your-api-key
```

5.1.3 验证连接

```
alb-cli status
```

5.2 全局参数

参数	说明	默认值
`--output`	输出格式：`json` 或 `table`	`json`
`-h, --help`	查看帮助	-

全局参数可放在子命令前后：

```
alb-cli --output table site list --type http
alb-cli site list --type http --output table
```

5.3 输出格式

5.3.1 JSON（默认）

所有查询命令默认输出 JSON，方便 Agent 解析。

```
alb-cli site list --type http
```

示例输出：

```
[
  {
    "id": 1,
    "name": "example",
    "domains": "example.com",
    "listen_port": 80,
    "enable_ssl": false,
    "status": "enabled"
  }
]
```

5.3.2 Table

适合人工阅读。

```
alb-cli site list --type http --output table
```

示例输出：

```
ID  NAME     PORT  STATUS   DOMAINS
1   example  80    enabled  example.com
```

5.3.3 错误输出

所有错误统一输出到 `stderr`：

```
missing API key: set ALB_API_KEY or ~/.alb/config.yaml api_key
```

服务端返回的错误也会直接打印消息。

5.4 命令参考

5.4.1 `site` — 站点管理

```
alb-cli site [子命令] --type http|stream [参数]
```

全局标志：

标志	说明	默认值
`--type`	站点类型：`http` 或 `stream`	`http`
`--status`	按状态过滤：`enabled` / `disabled`	-

`site list` — 列出站点

```
alb-cli site list --type http
alb-cli site list --type http --status enabled
alb-cli site list --type stream --output table
```

`site get` — 查看站点详情

```
alb-cli site get 1 --type http
```

`site create` — 从 JSON 文件创建站点

```
alb-cli site create --type http --file site.json
```

`site.json` 示例（HTTP）：

```
{
  "name": "example",
  "domains": "example.com www.example.com",
  "listen_port": 80,
  "enable_ssl": false,
  "locations": "
[{\"path\":\"/\",\"type\":\"proxy\",\"proxy_pass\":\"http://127.0.0.1:8
}
```

`site.json` 示例（Stream）：

```
{
  "name": "tcp-example",
  "listen_port": 9090,
  "protocol": "TCP",
  "backend_servers": "
[{\"address\":\"127.0.0.1:8080\",\"weight\":1}]"
}
```

`site update` — 从 JSON 文件更新站点

```
alb-cli site update 1 --type http --file site.json
```

JSON 文件中不需要包含 `id`，CLI 会自动将命令行传入的 ID 写入请求体。

`site enable` / `site disable` — 启用/禁用站点

```
alb-cli site enable 1 --type http
```

```
alb-cli site disable 1 --type stream
```

`site delete` — 删除站点

```
alb-cli site delete 1 --type http
```

`site bind-ssl` — 为 HTTP 站点绑定 SSL 证书

```
alb-cli site bind-ssl --site-id 1 --cert-id 2
```

5.4.2 `status` — ALB 运行状态

```
alb-cli status
```

返回当前 ALB 运行状态、监听端口、server 数量等。

5.4.3 `metrics` — 监控数据

`metrics traffic` — 流量趋势

```
alb-cli metrics traffic --metric-type min   # 过去 1 小时
alb-cli metrics traffic --metric-type hour  # 过去 24 小时
```

`metrics nodes` — 多节点状态快照

```
alb-cli metrics nodes
```

5.4.4 `config reload` — 重载 ALB 配置

```
alb-cli config reload
```

执行后 ALB 会重新加载配置文件，使站点变更生效。

5.4.5 `log` — 日志查看

`log list` — 日志文件列表

```
alb-cli log list
```

`log show` — 查看完整日志内容

```
alb-cli log show access.log
```

`log tail` — 查看日志尾部

```
alb-cli log tail access.log -n 100
```

5.5 安全策略

5.5.1 危险操作确认

以下命令默认需要终端交互确认：

- `site create`
- `site update`
- `site enable`
- `site disable`
- `site delete`
- `site bind-ssl`
- `config reload`

示例交互：

```
$ alb-cli config reload
reload ALB configuration? [y/N]: y
```

5.5.2 非交互环境

如果 `stdout` 或 `stdin` 不是 TTY，写操作会被拒绝：

```
destructive operation requires an interactive terminal
```

这可以避免 Agent 在 CI / 脚本中误触发破坏性操作。

5.6 典型工作流

5.6.1 发布一个 HTTP 站点

```
# 1. 创建站点
alb-cli site create --type http --file site.json

# 2. 重载配置使其生效
alb-cli config reload

# 3. 验证站点状态
alb-cli site list --type http --output table
```

5.6.2 为站点启用 HTTPS

```
# 1. 绑定证书
alb-cli site bind-ssl --site-id 1 --cert-id 2

# 2. 重载配置
alb-cli config reload
```

5.6.3 日常巡检

```
alb-cli status
alb-cli metrics traffic --metric-type min
alb-cli metrics nodes
```

5.6.4 故障排查

```
alb-cli log list
alb-cli log tail error.log -n 50
```

5.7 后端接口映射

CLI 命令	后端 API
site list --type http	GET /api/alb/console/external/v1/sites/http
site get 1 --type http	GET /api/alb/console/external/v1/sites/http/1
site create --type http	POST /api/alb/console/external/v1/sites/http
site update 1 --type http	PUT /api/alb/console/external/v1/sites/http
site enable 1 --type http	POST /api/alb/console/external/v1/sites/http/1/enable
site disable 1 --type http	POST /api/alb/console/external/v1/sites/http/1/disable
site delete 1 --type http	DELETE /api/alb/console/external/v1/sites/http?id=1
site bind-ssl	POST /api/alb/console/external/v1/sites/http/bind-ssl
status	GET /api/alb/console/external/v1/alb/status
metrics traffic	GET /api/alb/console/external/v1/alb/flow?type=min
metrics nodes	GET /api/alb/console/external/v1/analytic/nodes
config reload	POST /api/alb/console/external/v1/config/reload
log list	GET /api/alb/console/external/v1/alb/logger/name
log show	GET /api/alb/console/external/v1/alb/logger/text?logger_name=...

6 AI Agent 集成

ALB 提供面向 AI Agent（Claude、OpenCode 等）的标准 SKILL 描述，使 Agent 可以安全、可控地调用 ALB 管控台能力。

6.1 SKILL 用途

通过 `alb-cli` 与 ALB 管控台交互，AI Agent 可以完成：

- 站点管理（创建、更新、启用/禁用、删除、SSL 绑定）
- 状态监控（运行状态、流量趋势、节点健康度）
- 配置重载（修改后即时生效）
- 日志排查（list / show / tail）

6.2 SKILL安装

`【ALB安装路径】/utils/alb-cli/alb-cli/SKILL.md` 是面向 Claude Code、OpenCode 等 AI Agent 的 Skill 定义文件，安装后 Agent 可以通过自然语言直接调用 `alb-cli` 完成 ALB 运维任务。

6.2.1 在 Claude Code 中安装

Claude Code 会自动加载项目根目录 `.claude/skills/<skill-name>/SKILL.md` 下的 Skill。

6.2.1.1 方式一：复制到项目 Skill 目录（推荐）

```
# 从源码目录复制到项目根目录的 .claude/skills/
cp -r 【ALB安装路径】/utils/alb-cli/alb-cli/alb-cli .claude/skills/
```

复制后目录结构：

```
.claude/skills/
└── alb-cli/
    └── SKILL.md
```

6.2.1.2 方式二：手动创建

在项目根目录执行：

```
mkdir -p .claude/skills/alb-cli
cp ALB安装路径】/utils/alb-cli/alb-cli/alb-cli/SKILL.md
.claude/skills/alb-cli/SKILL.md
```

6.2.1.3 验证安装

重启 Claude Code 或在对话中输入：

```
/alb-cli 查看状态
```

如果 Claude Code 识别了该 Skill，会自动执行 `alb-cli status` 并返回 ALB 运行状态。

6.3 在其他 Agent 工具中使用

6.3.1 OpenCode

OpenCode 同样遵循 `.claude/skills/<skill-name>/SKILL.md` 的目录约定。按方式一复制后即可使用，触发词与 Claude Code 相同。

6.3.2 通用兼容工具

任何支持 Markdown Skill 定义的 Agent 工具，只要读取 `SKILL.md` 中的 `description` 和 `allowed-tools`，即可调用其中定义的命令。安装方式通常是：

1. 将 `cli/docs/alb-cli/SKILL.md` 复制到该工具指定的 skill 目录
2. 确保工具能够访问 `alb-cli` 二进制（已随管控台打包到 `console/tools/`，或单独构建到 `cli/alb-cli`）
3. 配置环境变量或 `~/.alb/config.yaml`

6.4 前置依赖

使用Skill 前，确保目标环境满足：

1. **ALB 管控台可访问**：配置 `ALB_CONSOLE_URL`，例如 `http://127.0.0.1:8887`
2. **API Key 有效**：配置 `ALB_API_KEY`，该 key 需与后端 `config/config.yaml` 中 `external.api_keys` 的某一项匹配

示例环境变量：

```
export ALB_CONSOLE_URL=http://127.0.0.1:8887
export
ALB_API_KEY=9489ebf25ac6664c242e0b2f2b406164d4175ae227bd7b09db762d66c58
```

或配置文件 `~/.alb/config.yaml`：

```
url: http://127.0.0.1:8887
api_key:
9489ebf25ac6664c242e0b2f2b406164d4175ae227bd7b09db762d66c58fa32b
```

6.5 触发方式

用户输入 `/alb-cli` 或类似指令时触发，例如：

- "帮我看看 ALB 当前状态"
- "列出所有 HTTP 站点"
- "创建一个新站点"
- "重载 ALB 配置"
- "查看最近 100 行 error.log"

6.6 可用命令速查

任务	命令
查看运行状态	`alb-cli status`
列出 HTTP 站点	`alb-cli site list --type http`
列出 Stream 站点	`alb-cli site list --type stream`
查看站点详情	`alb-cli site get <id> --type http`
创建站点	`alb-cli site create --type http --file site.json`
更新站点	`alb-cli site update <id> --type http --file site.json`
启用/禁用站点	`alb-cli site enable <id> --type http` / `alb-cli site disable <id> --type http`
删除站点	`alb-cli site delete <id> --type http`
绑定 SSL 证书	`alb-cli site bind-ssl --site-id <id> --cert-id <id>`

查看流量	`alb-cli metrics traffic --metric-type min`
查看节点状态	`alb-cli metrics nodes`
重载配置	`alb-cli config reload`
日志列表	`alb-cli log list`
查看日志	`alb-cli log show <name>`
跟踪日志	`alb-cli log tail <name> -n <lines>`

6.7 默认输出与解析

- 默认输出 `json`，Agent 应优先解析 JSON。
- 需要人工可读表格时可加 `--output table`。
- 错误信息会输出到 `stderr`。

6.8 安全约束

> *写操作需要终端确认。以下命令在执行时会提示 `[y/N]`：*

- `site create`
- `site update`
- `site enable`
- `site disable`
- `site delete`
- `site bind-ssl`
- `config reload`

如果当前环境不是交互式终端（stdout/stdin 不是 TTY），这些命令会失败并返回：

```
destructive operation requires an interactive terminal
```

Agent 在调用这些命令前，必须先向用户确认，再让用户在终端执行，或要求用户显式授权。

6.9 标准工作流

6.9.1 工作流 1：发布 HTTP 站点

```
# 1. 创建站点（用户确认后执行）
alb-cli site create --type http --file site.json

# 2. 重载配置（用户确认后执行）
alb-cli config reload

# 3. 验证
alb-cli site list --type http
```

6.9.2 工作流 2：启用 HTTPS

```
# 用户确认后执行
alb-cli site bind-ssl --site-id 1 --cert-id 2
alb-cli config reload
```

6.9.3 工作流 3：日常巡检

```
alb-cli status
alb-cli metrics traffic --metric-type min
alb-cli metrics nodes
```

6.9.4 工作流 4：故障排查

```
alb-cli status
alb-cli log list
alb-cli log tail error.log -n 50
```

6.10 JSON 文件模板

创建/更新 HTTP 站点时，JSON 文件示例：

```
{
  "name": "example",
```

```
  "domains": "example.com www.example.com",
  "listen_port": 80,
  "enable_ssl": false,
  "locations": "
[{\"path\":\"/\",\"type\":\"proxy\",\"proxy_pass\":\"http://127.0.0.1:8
}
```

Stream 站点示例：

```
{
  "name": "tcp-example",
  "listen_port": 9090,
  "protocol": "TCP",
  "backend_servers": "
[{\"address\":\"127.0.0.1:8080\",\"weight\":1}]"
}
```

Agent 生成 JSON 文件后，可调用：

```
alb-cli site create --type http --file /tmp/site.json
```

6.11 常见错误处理

错误	处理
`missing console URL`	提示用户配置 `ALB_CONSOLE_URL` 或 `~/.alb/config.yaml`
`missing API key`	提示用户配置 `ALB_API_KEY`
`invalid api key`	提示用户在管控台重新生成 API Key 并启用外部 API
`destructive operation requires an interactive terminal`	说明该命令需要用户确认，请用户手动执行或切换到交互终端
`alb状态模块未开启`	提示用户在管控台开启状态模块后再查询流量

6.12 最佳实践

1. **先查询再修改**：创建/更新/删除站点前，先 `site list` 确认现状。

2. **修改后重载**：站点变更类操作（create/update/enable/disable/delete/bind-ssl）后，建议执行 `config reload`。
3. **优先 JSON**：Agent 解析时使用 `--output json` 默认值。
4. **避免批量危险操作**：不要在没有用户确认的情况下循环执行 delete/disable/reload。
5. **文件清理**：临时 JSON 文件使用 `/tmp/` 目录，任务完成后可删除。

---

7 运维与监控接口

本节汇总 ALB 对外提供的运维与监控相关能力，便于运维平台和监控系统集成。

7.1 ALB 运行状态

接口/命令	说明
`GET /api/alb/console/external/v1/alb/status`	当前 ALB 节点运行状态、监听端口、server 数量
`alb-cli status`	CLI 入口，等同上述 API

7.2 流量与节点监控

接口/命令	说明
`GET /api/alb/console/external/v1/alb/flow?type=min`	过去 1 小时流量趋势
`GET /api/alb/console/external/v1/alb/flow?type=hour`	过去 24 小时流量趋势
`GET /api/alb/console/external/v1/analytic/nodes`	多节点状态快照（HA 集群下使用）
`alb-cli metrics traffic` / `alb-cli metrics nodes`	CLI 入口

> ***提示***：*调用 `metrics traffic` 前，请确认 ALB 状态模块（stub_status）已开启。*

7.3 配置重载

接口/命令	说明
`POST /api/alb/console/external/v1/config/reload`	通知 ALB 重新加载配置
`alb-cli config reload`	CLI 入口，会触发交互确认

> ***注意***：*HA 集群中写操作会自动从主节点同步到备节点，无需在备节点重复调用。*

7.4 日志查询

接口/命令	说明
`GET /api/alb/console/external/v1/alb/logger/name`	日志文件列表
`GET /api/alb/console/external/v1/alb/logger/text?logger_name=...`	读取指定日志内容

`alb-cli log list` / `alb-cli log show` / `alb-cli log tail`	CLI 入口

8 典型集成场景

8.1 场景 1：CI/CD 自动发布 HTTP 站点

在持续集成流水线中，通过 alb-cli 把构建产物发布到 ALB：

```
# 1. 准备站点 JSON
cat > /tmp/site.json <<EOF
{
  "name": "web-${BUILD_ID}",
  "domains": "web.example.com",
  "listen_port": 80,
  "locations": "
[{\"path\":\"/\",\"type\":\"proxy\",\"proxy_pass\":\"http://${APP_HOST}
  "auto_reload": true,
  "status": "enabled"
}
EOF

# 2. 由运维在终端执行（避免 CI 误触发）
export ALB_CONSOLE_URL=https://alb-master:8887
export ALB_API_KEY=${ALB_API_KEY}
alb-cli site create --type http --file /tmp/site.json
alb-cli config reload
```

8.2 场景 2：与 ERP / 业务系统对接

通过外部 API，在 ERP 上录入"业务系统"时自动在 ALB 创建反向代理：

```
import requests

API_KEY = "alb-erp-2024-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
BASE = "http://alb-master:8887/api/alb/console/external/v1"

def create_proxy(name, domain, port, upstream):
```

```
    return requests.post(
        f"{BASE}/sites/http",
        headers={"X-API-Key": API_KEY, "Content-Type":
"application/json"},
        json={
            "name": name,
            "domains": domain,
            "listen_port": port,
            "locations":
f'[{{"path":"/","type":"proxy","proxy_pass":"{upstream}"}}]',
            "auto_reload": True,
            "status": "enabled",
        },
    ).json()
```

8.3 场景 3：日常巡检机器人（AI Agent）

将 SKILL 接入 Claude / OpenCode，让 AI 进行日常巡检：

```
用户：帮我看下 ALB 现在的状态，并查看最近 50 行 error.log

AI 执行：
1. alb-cli status
2. alb-cli log list
3. alb-cli log tail error.log -n 50

AI 输出（结合 JSON 解析）：当前 ALB 主节点运行正常，监听 3 个端口，
最近 50 行 error.log 中未发现 ERROR 级别日志。
```

8.4 场景 4：故障自动定位

```
用户：192.168.1.100:8080 这个后端访问不了，帮我看看

AI 执行：
```

```
1. alb-cli site list --type http    (找到对应站点 ID)
2. alb-cli site get <id> --type http
3. alb-cli log tail error.log -n 100
4. alb-cli metrics nodes

AI 输出：站点 test-api 的 upstream 后端 192.168.1.100:8080 健康检查失败，
请检查后端服务是否存活。
```

---

9 集成建议

- **不要把 API Key 写进代码仓库**，建议通过环境变量、K8s Secret、Vault 等外部方式注入。
- **生产环境使用 HTTPS** 访问管控台，避免 API Key 在网络上明文传输。
- **为不同业务系统分配独立 API Key**，便于按需吊销、审计与配额隔离。
- **HA 集群下调用写操作前**，建议先 `alb-cli status` 确认主节点身份，再将请求发往主节点。

---

10 错误码速查表

10.1 外部 API 错误码

code	含义	建议处理
0	成功	-
7	通用业务错误	读取 `msg` 字段，根据业务提示处理
401	鉴权失败	检查 `X-API-Key` 是否正确、是否过期
403	外部 API 未启用	在管控台 **系统管理 > 开放 API** 中开启
10001	当前节点为 HA 备节点	将请求发往主节点

10.2 alb-cli 常见错误

错误信息	原因	建议处理
`missing console URL`	未配置 `ALB_CONSOLE_URL` 或 `~/.alb/config.yaml`	检查环境变量或配置文件
`missing API key`	未配置 `ALB_API_KEY`	检查环境变量或配置文件
`invalid api key`	API Key 错误或未启用外部 API	在管控台重新生成并启用
`destructive operation requires an interactive terminal`	非 TTY 环境下执行写操作	在终端中手动执行
`alb状态模块未开启`	调用 `metrics traffic` 时 ALB 状态模块未启用	在管控台开启 stub_status 或状态模块

11 最佳实践与故障排查

11.1 最佳实践

1. **先查询再修改**：写操作前先通过 `site list` 确认目标 ID 与当前状态。
2. **修改后重载**：所有站点变更类操作后调用 `config reload`。
3. **写操作走交互确认**：CI / 脚本中不要绕过 alb-cli 的安全确认，建议由运维在终端手工触发。
4. **使用 HTTPS**：生产环境通过 HTTPS 访问管控台，避免 API Key 明文传输。
5. **分级 API Key**：不同业务系统分配独立 API Key，并按需在管控台吊销。
6. **审计追溯**：所有外部 API 调用都会落入 `sys_operation_logs`，可通过 SQL 查询 `operator_name LIKE 'external:%'`。
7. **HA 集群**：写操作发往主节点；备节点写操作会被拒绝（code=10001）。

11.2 故障排查

现象	可能原因	建议处理
调用 API 返回 401	API Key 错误或过期	在管控台重新生成 API Key
调用 API 返回 403	外部 API 未启用	在管控台 **系统管理 > 开放 API** 中启用
写操作返回 10001	当前为 HA 备节点	将请求发往主节点
alb-cli 提示 missing api key	环境变量 / 配置文件未配置	配置 `ALB_CONSOLE_URL` 与 `ALB_API_KEY`
alb-cli 提示 requires an interactive terminal	非 TTY 环境	切换到终端或由运维手工执行
`metrics traffic` 提示状态模块未开启	ALB stub_status 未启用	在管控台开启状态模块
创建站点后访问 502	后端服务不可达	检查 upstream 节点连通性

12 版本更新说明

日期	手册版本	适用产品	更新说明
2026年06月	V2.0.6	ALB V2.0.6 标准版	全新发布：外部 API 接口、alb-cli、AI Agent SKILL

全国统一服务热线

4008-555-800

金蝶天燕云计算股份有限公司(简称“金蝶天燕云”)成立于2000年，前身为“金蝶中间件公司”，是金蝶集团旗下新一代软件基础云平台服务商，云计算国家标准制定企业，国家信创产业核心软件企业。金蝶天燕是国家863重点研发计划与核高基重大专项承接企业，也是“两网一站四库十二金”国家重点工程的基础平台提供商，产品广泛应用于政府、军工、金融、能源等关键行业，累计服务客户总数超过10万家。

Apusic 金蝶天燕

云计算国家标准制定企业

金蝶集团旗下基础软件企业

信息技术应用创新核心企业

官网：www.apusic.com