

# UAP 系统对接说明（面向自动化阅读）

本文档**独立成篇**，描述如何在业务系统中对接 **Unified Authentication Platform（UAP，统一认证平台）**。阅读者应能仅凭本文完成：配置项填写、登录、SSO、**用户查询与映射同步**、**消息发送**（系统通知与广播）。

## 示例约定

下文示例中：

- `https://api.hnyunzhu.com/api/v1` 表示 `UAP_API_BASE`（生产环境统一认证 API）；
- `app_demo_001`、`your_app_secret`、`eyJhbG...` 等为占位符，需替换为真实值；
- `sign` 为按第 4 节算法计算得到的 **64 位小写十六进制**字符串，示例里用 `deadbeef...` 形式表示，**不能**直接复制使用；
- 时间戳 `1710000000` 仅为示例，调用时需使用当前 Unix 秒。

## 1. 两类基础地址

变量名	含义	示例
UAP_API_BASE	<b>HTTP(S) API 根路径</b> ，必须包含版本前缀 <code>/api/v1</code>	<code>https://api.hnyunzhu.com/api/v1</code>
UAP_WEB_BASE	<b>用户浏览器访问的前端站点根</b> （用于跳转登录页；本部署与 API 同主机）	<code>https://api.hnyunzhu.com</code>

下文所有接口路径均相对于 `UAP_API_BASE` 拼接，例如：`UAP_API_BASE + /simple/login` → `https://api.hnyunzhu.com/api/v1/simple/login`。

跳转登录页使用 `UAP_WEB_BASE`：

- PC： `{UAP_WEB_BASE}/login?app_id=<APP_ID>`
- 移动端 H5： `{UAP_WEB_BASE}/mobile/login?app_id=<APP_ID>`

## 2. 对接前必填配置

在 UAP 管理后台创建应用后，请收集并安全保存以下信息：

配置项	说明
APP_ID	应用对外标识（字符串，如 <code>app_xxxxxxxx</code> ）
APP_SECRET	用于 <b>HMAC-SHA256 签名</b> ， <b>仅允许在业务服务端使用</b> ，禁止写入前端或客户端
APP_ACCESS_TOKEN	应用访问令牌，用于 <b>M2M 接口</b> （HTTP Header： <code>X-App-Access-Token</code> ）
REDIRECT_URIS	在应用中配置的回调地址列表；Redirect SSO、 <code>sso-login</code> 、 <code>exchange</code> 等流程依赖合法回调

可选：若使用标准 OIDC 协议，还需 OIDC 客户端凭据（本文不展开 OIDC 全流程）。

### 2.1 对接方填写（凭据留空）

实施对接时，请将贵司从 UAP 管理后台获取的值填入下表（可打印或复制后替换正文中的占位符）。**App Secret 仅保存在服务端，勿提交到代码仓库或前端**。下表中 **API / 前端根地址** 已按当前生产环境填写；若贵司使用独立域名部署，请改为实际地址。

项目	变量名（本文档）	填写
API 根地址	UAP_API_BASE	<code>https://api.hnyunzhu.com/api/v1</code>
前端站点根地址	UAP_WEB_BASE	<code>https://api.hnyunzhu.com</code>

项目	变量名（本文档）	填写
App ID	APP_ID	
App Secret	APP_SECRET	
Access Token（应用访问令牌）	APP_ACCESS_TOKEN	

说明：

- **App ID、App Secret**：用于 Simple Auth 签名、消息接口 X-App-Id / X-Sign 等。
- **Access Token**：用于 HTTP Header X-App-Access-Token，对应 M2M（用户搜索、映射同步等）；**不是** App Secret。
- **UAP\_API\_BASE** 必须包含版本前缀 `/api/v1`；**UAP\_WEB\_BASE** 为浏览器访问登录页所用站点根（本部署与 API 同主机时为 `https://api.hnyunzhu.com`）。

### 3. 认证方式总览

对接时会遇到**四类**身份校验，请勿混用：

方式	典型场景	请求形式
<b>A. JSON Body 签名（Simple Auth）</b>	<code>/simple/login</code> 、 <code>/simple/validate</code> 、 <code>/simple/exchange</code> 等	Body 内含 <code>timestamp</code> 、 <code>sign</code> ，用 <code>APP_SECRET</code> 对 <b>除 <code>sign</code> 外的</b> 参数做 HMAC-SHA256（见第 4 节）
<b>B. HTTP Header 签名（应用调消息接口）</b>	服务端代应用调用 <code>POST /messages/</code>	Header：X-App-Id、X-Timestamp、X-Sign（签名字符串仅含 <code>app_id</code> 与 <code>timestamp</code> ，见第 4.2 节）
<b>C. 用户 JWT</b>	用户已登录 UAP 后调用需身份的接口（如 <code>POST /auth/login/json</code> 、管理类接口等）	Authorization: Bearer <access_token>
<b>D. M2M 访问令牌</b>	用户搜索、映射同步、全量用户拉取	X-App-Access-Token: <APP_ACCESS_TOKEN>（部分场景亦支持应用 JWT Bearer，与平台实现一致）

### 4. 签名算法（HMAC-SHA256）

平台校验逻辑与实现一致：**参数键名按 ASCII 升序排列**，拼接为 `key1=value1&key2=value2`（不含 `sign`），再对拼接串做 **HMAC-SHA256**，密钥为 UTF-8 编码的 `APP_SECRET`，结果为小写 **十六进制**字符串。

**时间戳**：`timestamp` 为 Unix 秒级整数；服务端对时间偏差有约 **300 秒**容差。

#### 4.1 Simple Auth（JSON Body）

参与签名的键为请求 JSON 中**除 `sign` 以外的**所有键（`None` 值一般应排除，与常见实现一致）；**仅包含实际发送的字段**。各接口以下文「参与签名的字段」为准。

**Python 示例：**

```
import hmac
import hashlib

def sign_simple_auth(secret: str, params: dict) -> str:
    data = {k: v for k, v in params.items() if k != "sign" and v is not None}
    query_string = "&".join(f"{k}={data[k]}" for k in sorted(data.keys()))
    return hmac.new(
        secret.encode("utf-8"),
        query_string.encode("utf-8"),
        hashlib.sha256,
    ).hexdigest()
```

4.2 消息接口（Header 签名）

仅对两个键签名： app\_id （即 X-App-Id 的值）、 timestamp （即 X-Timestamp 的字符串值，与 Header 中一致）。拼接串为：

```
app_id=<APP_ID>&timestamp=<TIMESTAMP>
```

对该字符串做 HMAC-SHA256，结果放入 X-Sign 。

消息 Header 签名示例（伪代码）：

- 待签名字符串： app\_id=app\_demo\_001&timestamp=1710000000
- X-Timestamp 取值与字符串中的 timestamp **完全一致**（通常为数字的十进制字符串，如 "1710000000"）。

```
X-App-Id: app_demo_001
X-Timestamp: 1710000000
X-Sign: 1a2b3c4d5e6f708192a3b4c5d6e7f8090a1b2c3d4e5f678901234567890abc
```

5. 登录相关接口

5.1 POST /simple/login （密码登录）

- Content-Type**： application/json
- 说明**： 未提供 app\_id 时为**平台账号登录**（返回用户 JWT）；提供 app\_id 时为**应用 SSO 登录**（返回 ticket）。应用侧若提供 sign / timestamp，二者须**同时**出现，且服务端会校验签名（推荐服务端调用时始终带签名）。

请求体字段

字段	类型	必填	说明
app_id	string   null	否	不提供：平台登录；提供：应用 SSO，返回 ticket
identifier	string	是	用户标识：手机号、映射 mapped_key 或映射邮箱等（平台登录时按手机号查找）
password	string	是	登录密码
remember_me	boolean	否	默认 false；为 true 时 JWT 有效期延长（ <b>仅平台登录</b> 生效）
timestamp	int	条件	与 sign 同时提供或同时省略；提供时参与签名
sign	string	条件	HMAC-SHA256 十六进制； <b>应用 SSO 时推荐必带</b>

应用 SSO 且启用签名时，参与签名的字段为： app\_id、 identifier、 password、 timestamp （不含 remember\_me）。

响应 200： PasswordLoginResponse

字段	类型	何时出现	说明
ticket	string   null	提供 app_id 且登录成功	临时票据，供 POST /simple/validate 使用

字段	类型	何时出现	说明
access_token	string   null	未提供 app_id 且登录成功	用户 JWT
token_type	string   null	平台登录成功	固定为 bearer
role	string   null	平台登录成功	如 SUPER_ADMIN 、 DEVELOPER 、 ORDINARY_USER

应用 SSO 成功时典型返回： { "ticket": "<票据字符串>" } （无 access\_token ）。

## 调用与返回示例

### A. 应用 SSO（带签名，返回 ticket ）

```
POST https://api.hnyunzhu.com/api/v1/simple/login HTTP/1.1
Content-Type: application/json

{
  "app_id": "app_demo_001",
  "identifier": "13800138000",
  "password": "UserPassword123",
  "timestamp": 1710000000,
  "sign": "1a2b3c4d5e6f708192a3b4c5d6e7f8090a1b2c3d4e5f678901234567890abcd"
}
```

sign 由 APP\_SECRET 对以下键签名（不含 sign）： app\_id 、 identifier 、 password 、 timestamp 。

响应 200：

```
{
  "ticket": "TICKET-7f8e9d0a-1234-5678-abcd-ef0123456789"
}
```

### B. 平台登录（无 app\_id ，返回 JWT）

```
POST https://api.hnyunzhu.com/api/v1/simple/login HTTP/1.1
Content-Type: application/json

{
  "identifier": "13800138000",
  "password": "UserPassword123",
  "remember_me": false
}
```

响应 200：

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0In0.xxx",
  "token_type": "bearer",
  "role": "ORDINARY_USER"
}
```

错误示例（密码错误， 401 ）：

```
{
  "detail": "密码错误"
}
```

常见错误 HTTP 状态与 detail （节选）

状态码	含义（示例）
400	用户已禁用、签名无效（应用侧）
401	密码错误
404	用户未找到、应用未找到

5.2 POST /simple/sms-login （短信验证码登录）

- **Content-Type:** application/json
- **说明:** 逻辑与 5.1 对称——无 app\_id 返回平台 JWT；有 app\_id 返回 ticket。签名可选；若带 sign 必须同时带 timestamp。短信功能需在平台开启（否则可能 403）。

请求体字段

字段	类型	必填	说明
mobile	string	是	手机号
code	string	是	短信验证码
remember_me	boolean	否	默认 false；影响平台登录 JWT 有效期
app_id	string   null	否	不提供：平台登录；提供：应用 SSO，返回 ticket
timestamp	int	条件	与 sign 成对出现
sign	string	条件	应用 SSO 且启用签名时： <b>参与签名的字段为</b> app_id、mobile、code、timestamp

响应 200

与 5.1 相同，使用 PasswordLoginResponse（ticket 或 access\_token + token\_type + role）。

调用与返回示例

应用 SSO（带签名）

```
POST https://api.hnyunzhu.com/api/v1/simple/sms-login HTTP/1.1
Content-Type: application/json

{
  "app_id": "app_demo_001",
  "mobile": "13800138000",
  "code": "123456",
  "timestamp": 1710000000,
  "sign": "2b3c4d5e6f708192a3b4c5d6e7f8090a1b2c3d4e5f678901234567890abcde"
}
```

参与签名的键：app\_id、mobile、code、timestamp。

响应 200：

```
{
  "ticket": "TICKET-8a9b0c1d-2345-6789-bcde-f01234567890"
}
```

平台短信登录（无 app\_id）

```
{
  "mobile": "13800138000",
  "code": "123456",
  "remember_me": false
}
```

响应 200：

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..",
  "token_type": "bearer",
  "role": "ORDINARY_USER"
}
```

常见错误（节选）

状态码	含义（示例）
400	验证码错误或已过期、用户已禁用
403	短信登录功能未开启
404	用户未找到、应用未找到

5.3 POST /simple/validate（验证票据）

- **Content-Type:** application/json
- **认证:** 必须使用 **目标应用的 APP\_SECRET** 对 Body 签名（**sign / timestamp 必填**）。

请求体字段

字段	类型	必填	说明
ticket	string	是	来自登录跳转、exchange、sso-login 等流程的票据
app_id	string	是	<b>消费票据的应用 ID</b> （与为该应用签名的 Secret 一致）
timestamp	int	是	Unix 秒
sign	string	是	参与签名的字段为：ticket、app_id、timestamp

响应 200：TicketValidateResponse

票据有效时：

字段	类型	说明
valid	bool	true
user_id	int	平台用户主键
mobile	string	用户手机号
mapped_key	string   null	该用户在 <b>当前应用</b> 下的映射账号（无映射时为 null）
mapped_email	string   null	该用户在 <b>当前应用</b> 下的映射邮箱

票据无效或已消费：

字段	类型	说明
valid	bool	false

字段	类型	说明
user_id 、 mobile 等	—	一般为 null 或省略（以实际 JSON 为准）

调用与返回示例

参与签名的键：ticket 、 app\_id 、 timestamp （不含 sign ）。

```
POST https://api.hnyunzhu.com/api/v1/simple/validate HTTP/1.1
Content-Type: application/json

{
  "app_id": "app_demo_001",
  "ticket": "TICKET-7f8e9d0a-1234-5678-abcd-ef0123456789",
  "timestamp": 1710000001,
  "sign": "3c4d5e6f708192a3b4c5d6e7f8090a1b2c3d4e5f678901234567890abcdef"
}
```

响应 200 （成功）：

```
{
  "valid": true,
  "user_id": 1001,
  "mobile": "13800138000",
  "mapped_key": "ext_user_001",
  "mapped_email": "zhangsan@example.com"
}
```

响应 200 （票据无效或已使用）：

```
{
  "valid": false
}
```

错误示例（签名错误， 400 ）：

```
{
  "detail": "签名无效"
}
```

常见错误

状态码	含义（示例）
400	签名无效
404	应用未找到

5.4 平台用户登录（获取 JWT，非 Simple 票据流程）

POST /auth/login/json

- Content-Type: application/json

请求体：

字段	类型	必填	说明
mobile	string	是	用户手机号（平台账号）
password	string	是	密码
remember_me	boolean	否	默认 false ； true 时使用更长有效期 JWT
app_id	string   null	否	可选；直接平台登录通常可省略

响应 200：

字段	类型	说明
access_token	string	用户 JWT
token_type	string	固定为 bearer

常见错误： 400 （手机号或密码错误、账户已禁用）、 403 （账户待审核）。

## 调用与返回示例

POST /auth/login/json

POST https://api.hnyunzhu.com/api/v1/auth/login/json HTTP/1.1  
Content-Type: application/json

```
{
  "mobile": "13800138000",
  "password": "UserPassword123",
  "remember_me": true
}
```

响应 200：

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_type": "bearer"
}
```

POST /auth/login （表单）

POST https://api.hnyunzhu.com/api/v1/auth/login HTTP/1.1  
Content-Type: application/x-www-form-urlencoded

username=13800138000&password=UserPassword123

响应 200： 与 JSON 登录相同。

错误示例（ 400 ）：

```
{
  "detail": "手机号或密码错误"
}
```

（ POST /auth/login 表单字段与响应与上表一致，见上文 HTTP 示例。）



## 6. SSO 相关接口

### 6.1 浏览器跳转登录（Redirect SSO）

非 JSON 接口，步骤如下：

- 浏览器访问：`{UAP_WEB_BASE}/login?app_id=<APP_ID>` 或 `{UAP_WEB_BASE}/mobile/login?app_id=<APP_ID>`。
- 用户在 UAP 登录成功后，浏览器重定向到应用配置的**回调 URL**，查询参数携带 `ticket=<票据>`。
- 应用**服务端**调用 `POST /simple/validate`（见 5.3）换用户信息并建立会话。

**前提：**回调地址必须在应用 `REDIRECT_URIS` 中配置。

#### 调用与返回示例

浏览器打开登录页（仅说明 URL，无 JSON Body）：

`https://api.hnyunzhu.com/login?app_id=app_demo_001`

登录成功后，浏览器跳转到应用回调（示例）：

`https://biz.example.com/oauth/callback?ticket=TICKET-7f8e9d0a-1234-5678-abcd-ef0123456789`

业务后端再使用 **5.3** 的 `POST /simple/validate` 换用户信息。

### 6.2 POST /simple/sso-login（API 获取跳转 URL）

- Content-Type：** `application/json`
- 说明：**返回业务系统带 `ticket` 的完整 `redirect_url`，客户端应**导航**至该 URL。目标应用须存在；**SIMPLE\_API** 应用返回 `回调地址?ticket=...`；**OIDC** 应用返回简化后的根 URL（仅 `scheme + netloc`），详见下表。

#### 请求头（二选一）

模式	请求头 / Body
会话模式	<code>Authorization: Bearer &lt;用户 access_token&gt;</code> ；Body 仅需目标应用 <code>app_id</code>
凭据模式	无 Bearer；Body 须含 <code>app_id</code> 、 <code>username</code> 、 <code>password</code>

#### 请求体字段

字段	类型	必填	说明
<code>app_id</code>	<code>string</code>	是	要进入的 <b>目标应用 ID</b>
<code>username</code>	<code>string</code>	条件	未带有效 Bearer 时 <b>必填</b> ：手机号、映射 key 或映射邮箱等
<code>password</code>	<code>string</code>	条件	未带有效 Bearer 时 <b>必填</b>

**响应 200：** `SsoLoginResponse`

字段	类型	说明
<code>redirect_url</code>	<code>string</code>	<b>SIMPLE_API：</b> <code>{首个 redirect_uri}?ticket=&lt;票据&gt;</code> ； <b>OIDC：</b> 回调 URI 解析后的 <code>scheme://host[:port]</code> （无路径）

#### 常见错误

状态码	含义（示例）
400	应用未找到、未配置重定向 URI、用户已禁用、协议不支持等

状态码	含义（示例）
401	认证失败

调用与返回示例

会话模式（已持有用户 JWT）

```
POST https://api.hnyunzhu.com/api/v1/simple/sso-login HTTP/1.1
Content-Type: application/json
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

```
{
  "app_id": "target_app_002"
}
```

响应 200（SIMPLE\_API 应用）：

```
{
  "redirect_url": "https://biz.example.com/callback?ticket=TICKET-aaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee"
}
```

凭据模式（未登录）

```
POST https://api.hnyunzhu.com/api/v1/simple/sso-login HTTP/1.1
Content-Type: application/json
```

```
{
  "app_id": "target_app_002",
  "username": "13800138000",
  "password": "UserPassword123"
}
```

响应 200：同上，为带 ticket 的 redirect\_url。

6.3 POST /simple/exchange（源应用用户免登进目标应用）

- Content-Type: application/json
- 签名：使用源应用 APP\_SECRET；sign / timestamp 必填。

请求体字段

字段	类型	必填	说明
app_id	string	是	源应用 ID
target_app_id	string	是	目标应用 ID
user_mobile	string	是	用户在 UAP 的手机号（须已存在）
timestamp	int	是	Unix 秒
sign	string	是	参与签名：app_id、target_app_id、user_mobile、timestamp

响应 200：TicketExchangeResponse

字段	类型	说明
ticket	string	目标应用可用的票据

字段	类型	说明
redirect_url	string	目标应用首个 redirect_uri 拼接 ?ticket=<ticket> 的完整 URL（若未配置合法 URI，实现中可能回退占位，生产环境应保证配置正确）

常见错误

状态码	含义（示例）
400	签名无效
404	源应用 / 目标应用 / 用户未找到

目标应用收到 ticket 后，用**目标应用**的 app\_id + APP\_SECRET 调用 `POST /simple/validate`。

调用与返回示例

参与签名的键：app\_id、target\_app\_id、user\_mobile、timestamp（使用**源应用** Secret）。

```
POST https://api.hnyunzhu.com/api/v1/simple/exchange HTTP/1.1
Content-Type: application/json

{
  "app_id": "source_app_001",
  "target_app_id": "target_app_002",
  "user_mobile": "13800138000",
  "timestamp": 1710000002,
  "sign": "4d5e6f708192a3b4c5d6e7f8090a1b2c3d4e5f678901234567890abcdef01"
}
```

响应 200：

```
{
  "ticket": "TICKET-bbbbcccc-dddd-eeee-ffff-000011112222",
  "redirect_url": "https://target.example.com/callback?ticket=TICKET-bbbbcccc-dddd-eeee-ffff-000011112222"
}
```

6.4 GET /simple/sso/jump（通知内嵌 SSO 跳转）

用于消息中配置的跳转：用户点击后，若已在 UAP 登录则带上 Ticket 重定向到应用回调。

查询参数

参数	类型	必填	说明
app_id	string	是	目标应用 ID
redirect_to	string	是	登录成功后最终要到达的业务页 URL（经 URL 编码传入）

响应行为（非 JSON）

场景	HTTP	说明
当前 <b>未</b> 登录 UAP	307 / 302 等	RedirectResponse 到相对路径 /login?redirect=<当前完整 jump URL>（依赖网关/前端将用户导向登录页）
当前 <b>已</b> 登录	307 / 302 等	重定向到应用 redirect_uris 中首个 URI，查询参数包含 ticket、next（next 为 redirect_to）
应用不存在	404	detail 说明
未配置回调	400	detail 说明

调用与返回示例

请求 URL（redirect\_to 需 URL 编码）：

```
GET https://api.hnyunzhu.com/api/v1/simple/sso/jump?app_id=app_demo_001&redirect_to=https%3A%2F%2Foa.example.com%2Fapprove%2F123
```

已登录时：HTTP 重定向（302 / 307），Location 类似：

```
https://biz.example.com/callback?ticket=TICKET-xxx&next=https%3A%2F%2Foa.example.com%2Fapprove%2F123
```

未登录时：重定向到登录页，Location 可能类似：

```
/login?redirect=https%3A%2F%2Fapi.hnyunzhu.com%2Fapi%2Fv1%2Fsimple%2Fsso%2Fjump%3Fapp_id%3D...
```

7. 用户接口（M2M）

以下接口使用 X-App-Access-Token: <APP\_ACCESS\_TOKEN>（或实现所支持的应用 JWT），**不使用** Simple Auth 的 Body 签名。

7.1 推荐业务流程（查询 → 建用户 / 推映射）

对接方在「把业务账号与平台用户绑定」时，建议按下面顺序操作：

- 1. 先查询用户  
调用 GET /users/search（见 7.2），用手机号、姓名等关键词确认平台是否**已存在**该用户，并记录返回中的 id（**平台用户 ID**）、mobile、name 等。
  - **能查到**：说明平台已有账号，只需为本应用维护**映射**（mapped\_key、mapped\_email 等）。
  - **查不到**：需先在平台**创建用户**，再推映射。创建方式包括：
    - 在 **UAP 管理后台**人工新增；或
    - 若贵司具备超级管理员能力，调用管理端 POST /users/ 创建（需管理员权限）；或
    - 直接使用 POST /apps/mapping/sync 的 UPSERT：当手机号在平台不存在时，可在同一请求中携带 name 等，由平台**新建用户并建立映射**（见 7.3）。
- 2. 填充本应用账号信息并同步  
使用查询到的 mobile（及必要时 name）与业务侧 mapped\_key / mapped\_email，调用 POST /apps/mapping/sync：
  - **新增绑定**：sync\_action 为 UPSERT（默认），平台会插入或更新**当前应用**下的映射；若本次同时新建了平台用户，响应中 new\_user\_created 为 true。
  - **仅改映射**：对已存在用户再次 UPSERT，可更新 mapped\_key、mapped\_email、is\_active 等；**不可**通过该接口修改已存在用户的姓名、手机号等基础档案（与接口校验一致）。
  - **解除映射**：sync\_action 为 DELETE（仅删映射，不删平台用户）。
- 3. 全量对账（可选）  
需要批量拉取平台用户基础信息时，使用 GET /apps/mapping/users（见 7.4）。

7.2 GET /users/search（用户查询）

- **说明**：按关键词搜索**已激活、未删除**的平台用户，用于在推映射前确认人选。支持 **应用访问令牌** 或用户 JWT。
- **路径**：GET /users/search
- **认证**：X-App-Access-Token: <APP\_ACCESS\_TOKEN>，或 Authorization: Bearer <用户 JWT>

查询参数

参数	类型	必填	默认值	说明
q	string	否	—	关键词；对 <b>手机号、姓名、英文名</b> 模糊匹配（ilike）。不传时返回一批活跃用户（受 limit 限制）
limit	int	否	20	最大返回条数

说明：若使用用户 JWT 调用，结果中会排除当前登录用户本人；使用 应用令牌 时无此排除。

响应 200

返回 JSON 数组，元素为平台用户对象，主要字段如下：

字段	类型	说明
id	int	平台用户 ID（可用于发消息时的 receiver_id 等）
mobile	string	手机号
name	string   null	姓名
english_name	string   null	英文名
status	string	如 ACTIVE
role	string   null	角色
created_at	string	创建时间 ISO8601
updated_at	string	更新时间 ISO8601
is_deleted	int	是否删除标记

调用与返回示例

```
GET https://api.hnyunzhu.com/api/v1/users/search?q=13800138000&limit=20 HTTP/1.1
X-App-Access-Token: pA9s8d7f6g5h4j3k2l1m0n9o8p7q6r5s4t3u2v1w0
```

响应 200 示例：

```
[
  {
    "id": 1001,
    "mobile": "13800138000",
    "name": "张三",
    "english_name": "zhangsan",
    "status": "ACTIVE",
    "role": "ORDINARY_USER",
    "created_at": "2025-01-10T08:00:00",
    "updated_at": "2025-06-01T12:00:00",
    "is_deleted": 0
  }
]
```

7.3 POST /apps/mapping/sync

- Content-Type: application/json

请求体字段 UserSyncRequest

字段	类型	必填	说明
mobile	string	是	中国大陆手机号，正则 ^1[3-9]\d{9}\$ （与平台校验一致）
name	string   null	条件	UPSERT 时必填（非空姓名）；已存在用户时不可通过此接口改姓名等基础信息；DELETE 可不依赖姓名逻辑（以 sync_action 分支为准）
english_name	string   null	否	已废弃于 M2M；英文名由平台按规则生成

字段	类型	必填	说明
password	string   null	否	按平台实现（若有）
status	string   null	否	按平台实现
mapped_key	string	是	外部系统用户 ID，长度 1~100
mapped_email	string   null	否	外部邮箱
is_active	boolean   null	否	映射是否启用； null 表示不修改
sync_action	string   null	否	UPSERT （默认）或 DELETE

响应 200： MappingResponse

字段	类型	说明
id	int	映射记录 ID
app_id	int	应用数据库主键（整数）
user_id	int	平台用户 ID
mapped_key	string   null	映射账号
mapped_email	string   null	映射邮箱
user_mobile	string	用户手机号
user_status	string	用户统一状态描述
is_active	bool	映射是否有效；DELETE 后可能为 false
new_user_created	bool	是否本次新建了平台用户
generated_password	string   null	若平台为新用户生成了初始密码，可能返回

常见错误

状态码	含义（节选）
400	参数非法、与业务规则冲突（如重复映射、禁止修改字段）
403	Token 无效

调用与返回示例

UPSERT

```
POST https://api.hnyunzhu.com/api/v1/apps/mapping/sync HTTP/1.1
Content-Type: application/json
X-App-Access-Token: pA9s8d7f6g5h4j3k2l1m0n9o8p7q6r5s4t3u2v1w0
```

```
{
  "mobile": "13800138000",
  "name": "张三",
  "mapped_key": "ext_user_1001",
  "mapped_email": "zhangsan@partner.com",
  "is_active": true,
  "sync_action": "UPSERT"
}
```

响应 200：

```
{
  "id": 501,
  "app_id": 12,
  "user_id": 1001,
  "mapped_key": "ext_user_1001",
  "mapped_email": "zhangsan@partner.com",
  "user_mobile": "13800138000",
  "user_status": "ACTIVE",
  "is_active": true,
  "new_user_created": true,
  "generated_password": null
}
```

DELETE（仅删除映射）

```
POST https://api.hnyunzhu.com/api/v1/apps/mapping/sync HTTP/1.1
Content-Type: application/json
X-App-Access-Token: pA9s8d7f6g5h4j3k2l1m0n9o8p7q6r5s4t3u2v1w0
```

```
{
  "mobile": "13800138000",
  "mapped_key": "ext_user_1001",
  "sync_action": "DELETE"
}
```

响应 200：结构同 MappingResponse，is\_active 可能为 false（见字段表）。

7.4 GET /apps/mapping/users

- 认证：X-App-Access-Token

查询参数

参数	类型	必填	默认值	说明
skip	int	否	0	跳过条数
limit	int	否	100	每页条数

响应 200：UserSyncList

字段	类型	说明
total	int	符合条件的用户总数
items	array	用户列表

items[] 元素 UserSyncSimple：

字段	类型	说明
mobile	string	手机号
name	string   null	姓名
english_name	string   null	英文名

调用与返回示例

GET https://api.hnyunzhu.com/api/v1/apps/mapping/users?skip=0&limit=100 HTTP/1.1  
X-App-Access-Token: pA9s8d7f6g5h4j3k2l1m0n9o8p7q6r5s4t3u2v1w0

响应 200：

```
{
  "total": 1250,
  "items": [
    {
      "mobile": "13800138000",
      "name": "张三",
      "english_name": "zhangsan"
    },
    {
      "mobile": "13900139000",
      "name": "李四",
      "english_name": "lisi"
    }
  ]
}
```

8. 消息发送接口（仅 POST /messages/）

本节仅说明由应用服务端调用、使用 Header 签名（第 3 节方式 B、第 4.2 节）发送 NOTIFICATION 的两种形态：

- 1. 单用户系统通知（指定一名接收者）
- 2. 广播（向全体活跃用户各发一条）

不涉及用户间私信（type: MESSAGE）及用户 JWT 代发，对接方无需关注。

- 路径：POST /messages/
- Content-Type：application/json

8.1 请求头（应用签名）

Header	说明
Content-Type	application/json
X-App-Id	应用字符串 ID，与 APP_ID 一致
X-Timestamp	Unix 秒（与签名、Body 中逻辑一致）
X-Sign	对 app_id=<X-App-Id>&timestamp=<X-Timestamp> 的 HMAC-SHA256 十六进制

content\_type 取值说明

与后端枚举一致，发送时传字符串（大写）。未传时默认 TEXT。

取值	含义	content 建议形态
TEXT	普通文本通知	字符串
IMAGE	图片	对象存储中的 Object Key 或经平台处理的资源标识；若误传完整 URL，服务端可能尝试抽取 Key
VIDEO	视频	同上



取值	含义	content 建议形态
FILE	文件	同上
USER_NOTIFICATION	业务/申请类通知（会话列表中突出标题、正文可结构化）	字符串或 JSON 对象；传对象时服务端会序列化为 JSON 字符串存储，客户端常对 content 做 JSON.parse 解析

与 auto\_sso 的关系：当 auto\_sso=true 且已提供 target\_url、应用 app\_id 时，只要 type 为 NOTIFICATION 或 content\_type 为 USER\_NOTIFICATION，平台都会按同一规则生成 SSO jump 形式的 action\_url（见下文示例）。

选用建议：常规系统通知用 TEXT；需要「申请单样式」、富文本结构或前端自定义渲染时，可改用 USER\_NOTIFICATION 并在 content 中传结构化数据。

## 8.2 单用户系统通知（type: NOTIFICATION，非广播）

向一名接收者推送。接收人二选一指定：

- receiver\_id：平台用户 ID（整数）；或
- app\_id + app\_user\_id：应用字符串 ID + 该用户在本应用映射中的外部账号。

字段	类型	必填	说明
type	string	是	固定 NOTIFICATION
title	string	是	标题，最大 255 字
content	string   object	是	正文；content_type 为 USER_NOTIFICATION 时可传对象，见 8.1 content_type 取值说明
content_type	string	否	默认 TEXT；可选：TEXT、IMAGE、VIDEO、FILE、USER_NOTIFICATION（见 8.1）
receiver_id	int	条件	与 app_id + app_user_id 二选一
app_id	string	条件	与 app_user_id 同时出现时用于解析接收人
app_user_id	string	条件	接收方在本应用映射中的 ID
action_url	string   null	否	自定义跳转；若 auto_sso 为 true 可能被平台改写
action_text	string   null	否	按钮文案
target_url	string   null	否	auto_sso=true 时作为 SSO 目标页，用于生成 jump 链接
auto_sso	boolean	否	默认 false；为 true 时可生成 /api/v1/simple/sso/jump?... 形式的 action_url
sender_app_user_id	string   null	否	可选，标识应用侧「发起人」映射，用于审计展示

约束：is\_broadcast 为 false 或不传；须提供 receiver\_id 或 (app\_id + app\_user\_id)。

请求 / 响应示例（按 app\_user\_id 指定接收人，含 SSO 跳转）

```
POST https://api.hnyunzhu.com/api/v1/messages/ HTTP/1.1
Content-Type: application/json
X-App-Id: app_demo_001
X-Timestamp: 1710000000
X-Sign: 1a2b3c4d5e6f708192a3b4c5d6e7f8090a1b2c3d4e5f678901234567890abcd
```

```
{
  "app_id": "app_demo_001",
  "app_user_id": "ext_user_1001",
  "type": "NOTIFICATION",
  "content_type": "TEXT",
  "title": "审批提醒",
  "content": "您有一条待办审批",
  "auto_sso": true,
  "target_url": "https://oa.example.com/approve/123",
  "action_text": "去处理"
}

{
  "id": 90001,
  "sender_id": null,
  "receiver_id": 1001,
  "app_id": 12,
  "app_name": "演示应用",
  "type": "NOTIFICATION",
  "content_type": "TEXT",
  "title": "审批提醒",
  "content": "您有一条待办审批",
  "action_url": "/api/v1/simple/sso/jump?app_id=app_demo_001&redirect_to=https%3A%2F%2Foa.example.com%2Fapprove%2F123",
  "action_text": "去处理",
  "is_read": false,
  "created_at": "2026-04-07T10:00:00",
  "read_at": null
}
```

**补充示例（用平台用户 ID 指定接收人： receiver\_id ）**

```
POST https://api.hnyunzhu.com/api/v1/messages/ HTTP/1.1
Content-Type: application/json
X-App-Id: app_demo_001
X-Timestamp: 1710000001
X-Sign: <按 X-App-Id 与 X-Timestamp 计算的签名>
```

```
{
  "type": "NOTIFICATION",
  "content_type": "TEXT",
  "title": "到账提醒",
  "content": "您的订单已支付成功",
  "receiver_id": 1001
}
```

响应体结构与上一示例相同，字段值随请求变化（如 receiver\_id 为 1001）。

**补充示例（ content\_type: USER\_NOTIFICATION ，结构化 content + SSO ）**

POST https://api.hnyunzhu.com/api/v1/messages/ HTTP/1.1  
Content-Type: application/json  
X-App-Id: app\_demo\_001  
X-Timestamp: 1710000002  
X-Sign: <按 X-App-Id 与 X-Timestamp 计算的签名>

```
{
  "app_id": "app_demo_001",
  "app_user_id": "ext_user_1001",
  "type": "NOTIFICATION",
  "content_type": "USER_NOTIFICATION",
  "title": "请假申请待审批",
  "content": {
    "applicant": "张三",
    "days": 3,
    "reason": "年假"
  },
  "auto_sso": true,
  "target_url": "https://oa.example.com/hr/leave/999",
  "action_text": "去审批"
}
```

**响应 200（节选）：** content 存为 JSON 字符串； content\_type 为 USER\_NOTIFICATION ； action\_url 在 auto\_sso 为 true 时同样可生成 jump 链接。

```
{
  "type": "NOTIFICATION",
  "content_type": "USER_NOTIFICATION",
  "title": "请假申请待审批",
  "content": "{\"applicant\":\"张三\",\"days\":3,\"reason\":\"年假\"}",
  "action_url": "/api/v1/simple/sso/jump?app_id=app_demo_001&redirect_to=..."
}
```

### 8.3 广播（全员系统通知）

仅应用可调；向平台内**全部活跃用户**各投递一条通知。

字段	类型	必填	说明
is_broadcast	boolean	是	必须为 true
type	string	是	必须为 NOTIFICATION
title	string	是	标题
content	string   object	是	正文
content_type	string	否	默认 TEXT ；可选值见 8.1 （ USER_NOTIFICATION 等）
auto_sso	boolean	否	是否对跳转链接做 SSO 封装
target_url	string   null	否	与 auto_sso 配合使用
action_url	string   null	否	自定义跳转（不走 SSO 时）
action_text	string   null	否	按钮文案

**约束：**不要传 receiver\_id 、 app\_user\_id ；广播仅支持通知类型。

**请求示例**

POST https://api.hnyunzhu.com/api/v1/messages/ HTTP/1.1  
Content-Type: application/json  
X-App-Id: app\_demo\_001  
X-Timestamp: 1710000003  
X-Sign: 5e6f708192a3b4c5d6e7f8090a1b2c3d4e5f678901234567890abcdef0123

```
{
  "is_broadcast": true,
  "type": "NOTIFICATION",
  "content_type": "TEXT",
  "title": "系统维护",
  "content": "今晚 22:00-24:00 维护"
}
```

**响应 200：** 返回 `MessageResponse`，与「为全体用户各创建一条通知」中的**第一条**记录对应（便于对接方拿到一条结构化回包）；其余用户的消息结构相同，仅 `id`、`receiver_id` 等不同。若平台无活跃用户则返回 `400`，`detail` 为「没有可发送的活跃用户」。

```
{
  "id": 90010,
  "sender_id": null,
  "receiver_id": 1001,
  "app_id": 12,
  "app_name": "演示应用",
  "type": "NOTIFICATION",
  "content_type": "TEXT",
  "title": "系统维护",
  "content": "今晚 22:00-24:00 维护",
  "action_url": null,
  "action_text": null,
  "is_read": false,
  "created_at": "2026-04-07T10:00:00",
  "read_at": null
}
```

8.4 成功响应 MessageResponse （发送接口共性）

字段	类型	说明
id	int	消息 ID
sender_id	int   null	发送方用户 ID；应用系统通知可能为 null
receiver_id	int	接收方用户 ID
app_id	int   null	应用数据库主键（整数）
app_name	string   null	应用名称
type	string	本文档场景下为 NOTIFICATION （平台尚存在其他类型，见 OpenAPI）
content_type	string	内容类型：TEXT、IMAGE、VIDEO、FILE、USER_NOTIFICATION
title	string	标题
content	string	正文
action_url	string   null	跳转链接（系统通知可能存在）
action_text	string   null	按钮文案
is_read	bool	是否已读

字段	类型	说明
created_at	datetime	创建时间 ISO8601
read_at	datetime   null	已读时间

返回示例（200 OK，单条系统通知，与上文发送示例对应）：

```
{
  "id": 90001,
  "sender_id": null,
  "receiver_id": 1001,
  "app_id": 12,
  "app_name": "演示应用",
  "type": "NOTIFICATION",
  "content_type": "TEXT",
  "title": "审批提醒",
  "content": "您有一条待办审批",
  "action_url": "/api/v1/simple/sso/jump?app_id=app_demo_001&redirect_to=https%3A%2F%2Foa.example.com%2Fapprove%2F123",
  "action_text": "去处理",
  "is_read": false,
  "created_at": "2026-04-07T10:00:00",
  "read_at": null
}
```

（广播成功时结构相同，通常 receiver\_id 为本次作为「代表返回」的那条消息对应的用户 ID；action\_url、app\_id 等以实际响应为准。）

## 9. 调试与规范来源

- **交互式 API 文档：** <https://api.hnyunzhu.com/api/v1/docs>（Swagger）。
- 若本文与线上部署不一致，以**实际 OpenAPI 与接口返回**为准。

## 10. 安全要点（摘要）

1. APP\_SECRET 仅留在服务端；任何含 Secret 的签名不得在浏览器完成。
2. 区分 UAP\_API\_BASE 与 UAP\_WEB\_BASE。
3. APP\_ACCESS\_TOKEN 与 JWT 同样属于敏感凭据。
4. SSO 与 Ticket 流程务必使用 HTTPS 与合法回调域。

文档版本：与当前仓库后端 Schema/路由一致时，API 前缀为 /api/v1。