

# P2A Server API 说明文档

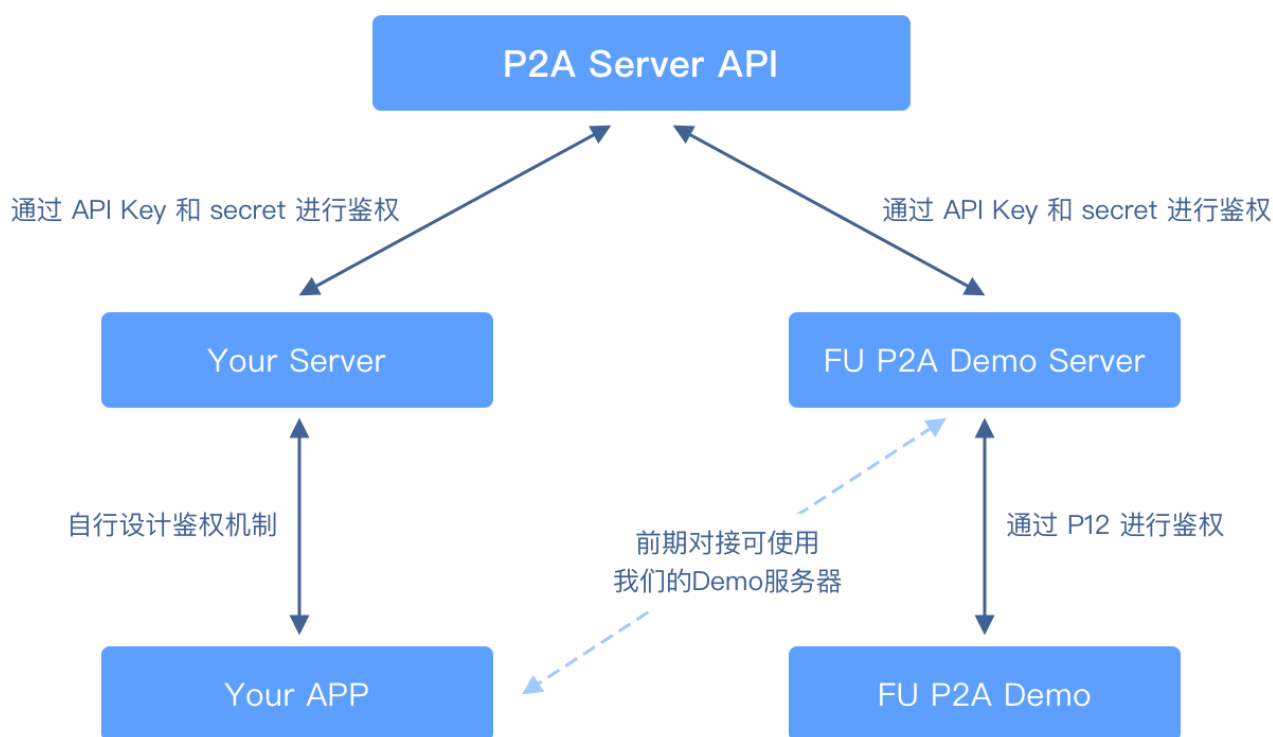
## 简介

P2A Server API的使用分为四个流程：

- 通过我司的WebAPI 控制台获取key和secret
- 联系我司，为上一步中的key添加P2A接口权限
- 通过key和secret获取access\_token
- 通过access\_token访问API

**注意：**请妥善保管key和secret。为了防止泄露，您需要在服务端调用P2A Server API，然后在客户端调用您的服务端接口

P2A Server、我们的测试服务器、我们的P2A Demo及您正式服务器和正式APP之间的架构关系如下图所示。

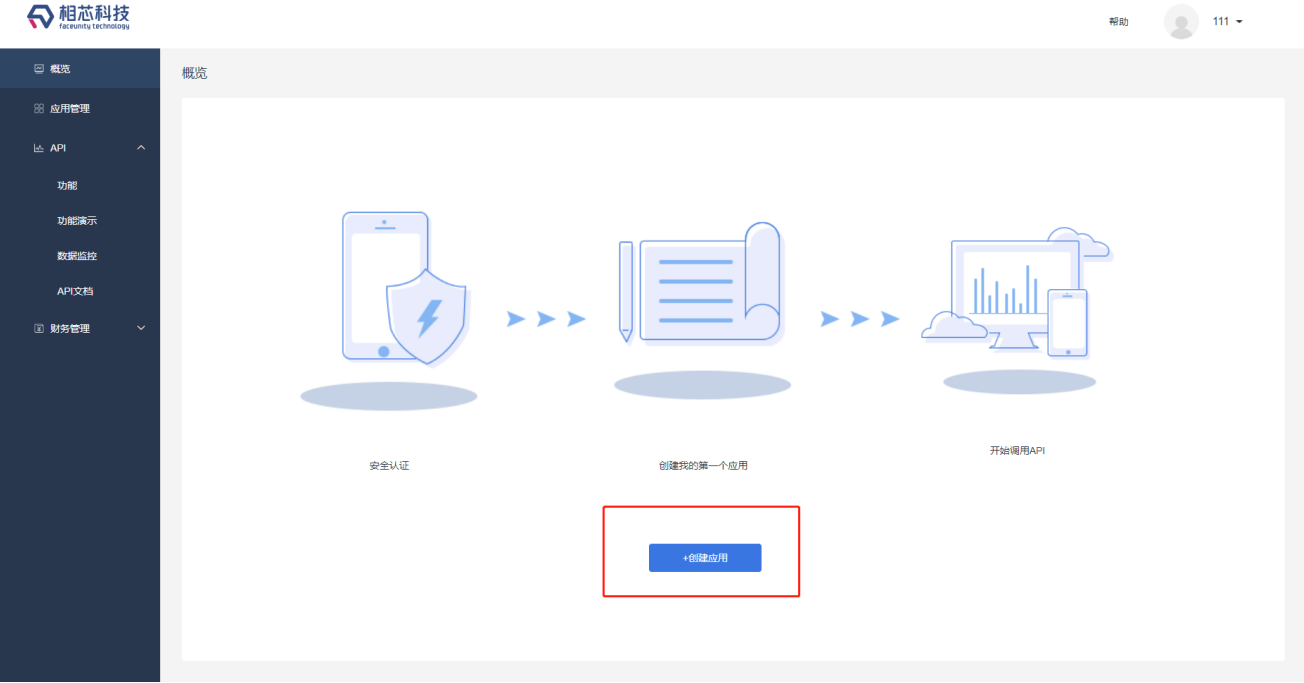


## 获取key和secret

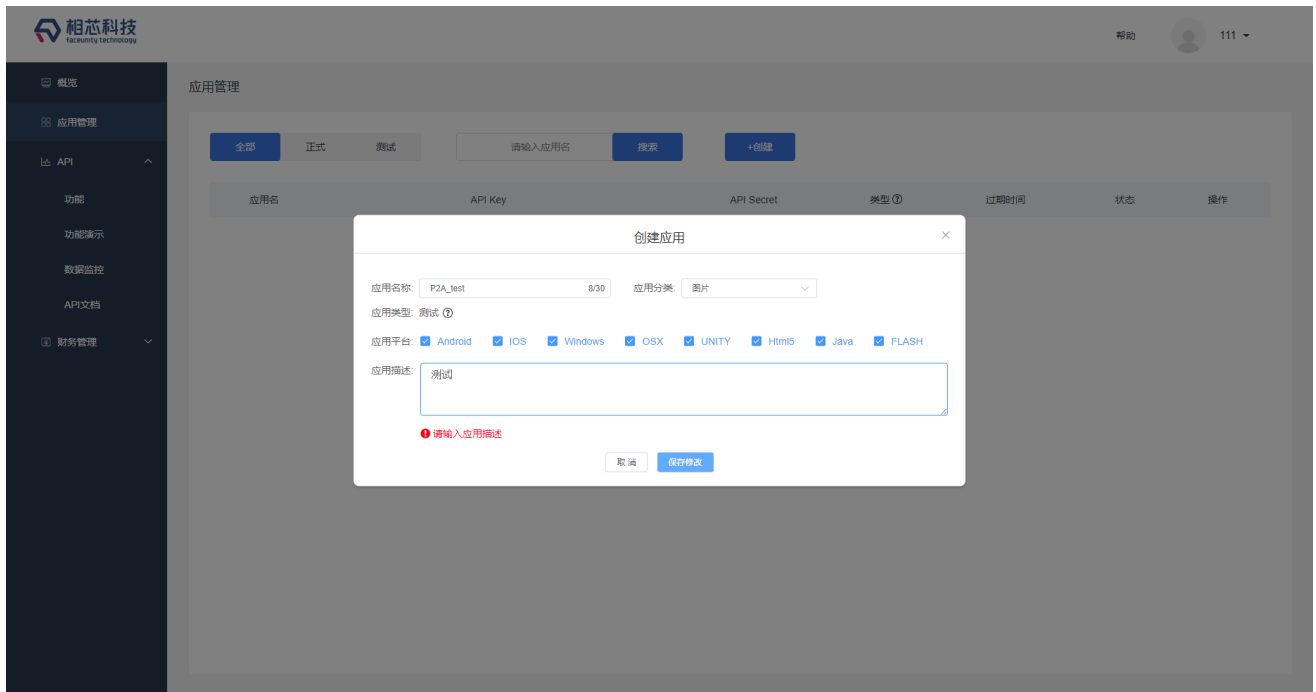
WebAPI控制台链接为：<https://console.faceunity.com>。新用户可自行注册相关账号。



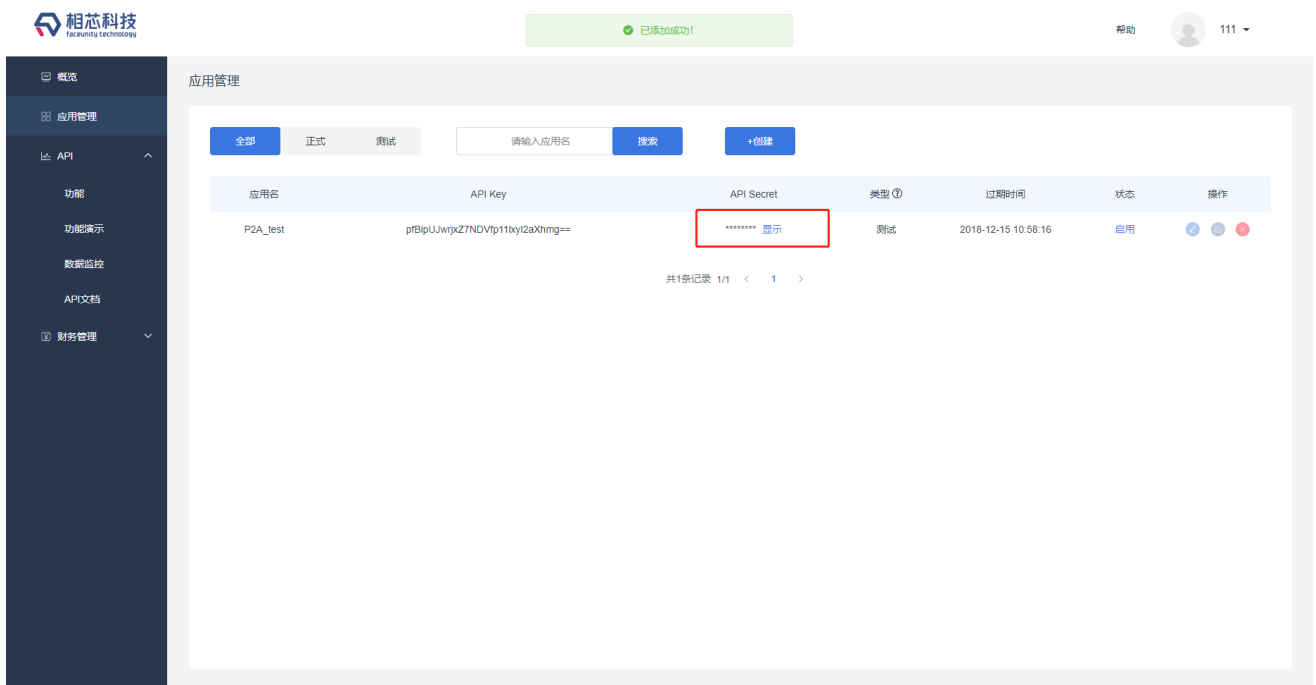
进入控制台后，选择创建应用。如下图所示。



输入应用的详细信息。其中应用名称，应用平台，以及应用描述为必填项目。



点击显示API Secret 即可以查看应用对应的Secret。



## 获取access\_token

获取access\_token时，需要用到key和secret。access\_token 用于后续的api访问。本文中假设

```
Key = '12345' //必须携带
Secret = '54321'
```

假设获取access\_token的请求链接为：<https://token.faceunity.com/api/v1/GetAccessToken?params=test&Key=12345>。获取access\_token的具体流程如下：

**\*\* 1. 将本次请求参数按照名称进行升序排列 \*\***

排序前：

```
{
  "params": "test",
  "Key": "12345"
}
```

排序后：

```
{
  "Key": "12345"
  "params": "test"
}
```

**\*\* 2. 构造被签名参数串 \*\***

被签名串的构造规则为：被签名串 = 所有请求参数拼接(无需HTTP转义)，并在本签名串的结尾拼接secret。

本例中排序后参数见上，故参数拼接后为 `Key12345paramstest`，然后加上账户中的secret，即本例中的 `54321`，最终被签名串为 `Key12345paramstest54321`。

**\*\* 3. 计算签名 \*\***

计算被签名串的sha1值生成Signature。

Nodejs代码

```
const { createHash } = require("crypto");

function signature(params, secret) {
  let keys = Object.keys(params).sort(); //按参数排序
  let sign_str: string = '';
  keys.forEach((value, index) => {      //构造被签名串
    sign_str += value + params[value];
  });
  sign_str += secret;                   //连接secret
  return createHash('sha1').update(sign_str).digest('hex'); //返回被签名串的sha1值
}
```

Python代码

```
import hashlib
import urlparse
import urllib

def _verfy_ac(private_key, params):
    items=params.items()
    # 请求参数串
    items.sort()
```

```
# 将参数串排序

params_data = "";
for key, value in items:
    params_data = params_data + str(key) + str(value)
params_data = params_data + private_key

sign = hashlib.sha1()
sign.update(params_data)
signature = sign.hexdigest()

return signature
# 生成的Signature值
```

#### \*\* 4. 拼接出请求链接 \*\*

将计算出的Signature拼接到尾部(各参数需要url编码,例如Key为lbA2MypNve2PeZpaOiPUGnSt+FHePw==,编码后为lbA2MypNve2PeZpaOiPUGnSt%2BFHePw%3D%3D)例如

```
https://token.faceunity.com/api/v1/GetAccessToken?
params=test&Key=12345&Signature=cac49742c5e52e63b285b6a549c7d362b19aa054
```

#### \*\* 5. 发送请求，获取access\_token \*\*

请求成功后会返回如下结构

```
{
  "code":2,
  "message":"success",
  "data":{
    "access_token":"82f205d0-8a31-11e8-8c11-b74c5a2e235c",
    "expirein":600
  }
}
```

data.access\_token:申请到的access\_token。

data.expirein:token的过期时间，单位为秒。

## 访问API

请求API时需在url地址或body中带上access\_token参数，如 `https://api.faceunity.com/api/p2a?access_token=82f205d0-8a31-11e8-8c11-b74c5a2e235c`。

## API

### /api/p2a

描述：P2A接口

请求方式：POST

参数：

参数名	数据类型	是否必须	说明
image	Binary	是	图片数据（小于2M），用于生成Avatar的人脸图片
gender	Number	是	性别参数，0为男性，1为女性

返回值:

参数名	数据类型	说明
code	Number	请求结果，0为请求失败，1为请求不合法，2为成功
message	String	返回信息
data	String	P2A API处理结果，即server.bundle,base64编码后的字符串

Demo(Node.js):

```
const request = require('request');
const fs = require('fs');

var formData = {
  image: fs.createReadStream('./test.jpg'),
  gender: 1
}

const access_token = "xxxx";

async function submit(access_token, data) {
  return new Promise((resLove, reject) => {
    request.post({
      url: `https://api.faceunity.com/api/p2a?access_token=${access_token}`,
      formData: data
    }, function optionalCallback(err, res) {
      if (err) {
        reject(err);
      }
      resLove(JSON.parse(res.body))
    })
  })
}

(async () => {
  try {
    const result = await submit(access_token, formData);
    console.log(result);
    fs.writeFileSync('./server.bundle', Buffer.from(result.data, "base64"));
  } catch (e) {
    console.log('Submit Error:', e);
  }
})();
```

之后可以将server.bundle传给P2A SDK进行处理。P2A SDK 相关的说明参见《P2A 对接说明文档》