云闪付 APP 授权登录指引

本文档展示了使用云闪付客户端 sdk 快速接入 app，使用云闪付用户授权登录，实现 app 与云闪付对接。
注意：本文档中的 DEMO 和 API 针对普通场景适用，如有疑问，联系中国银联云闪付事业部开放支持团队。

1. 产品介绍

APP 云闪付授权登录产品可以向云闪付合作商家 APP 或者行业机构 APP 输出云闪付用户信息产品，签署合作协议后可以申请集成云闪付快登产品 SDK，以适应合作机构 APP 云闪付用户授权登录需求。
云闪付授权登录产品基于云闪付客户端 sdk 和云闪付开放平台信息接口组成，基于 oauth2.0 授权协议，后台配置合作商家授权级别和可授权内容。基本流程如下(图中第三方 app 即为商家或行业机构 app)：


用户
第三方app
第三方后台
云闪付sdk
云闪付开放平台
云闪付app
云闪付后台
1: 用户打开第三方app，使用云闪付联登()
2: 点击云闪付授权联登()
2.1: 判断是否安装云闪付，有则打开，无则打开H5登录()
3: 展示授权页给用户()
4: 用户确认授权或登录()
4.1: 提交授权业务处理()
5: 处理成功返回()
5.1: 重定向第三方app并返回授权码auth_code()
6: 第三方app将授权码auth_code传给后台()
6.1: 调用云闪付开放平台接口用auth_code换取accessToken和openId()
7: 返回accessToken，openId()
8: 使用accessToken调用接口获取用户信息()
9: 返回用户授权相关信息()
10: 用户信息关联或注册新用户返回()
11: 返回登录用户信息()

1. 用户打开商家 app，点击云闪付第三方登录；
2. 商家 app 调用集成在商家 app 中云闪付 sdk 的接口，唤起云闪付 app(安装云闪付 app 的用户)或者打开 H5 云闪付登录页(未安装云闪付 app 的用户)并发送授权请求；
3. 云闪付返回授权页面，用户点击授权后云闪付重定向到商家 app，云闪付 sdk 返回给商家 app 授权码或者授权失败错误信息；
4. 商家 app 将授权码 code 发送给商家后台，商家后台用授权码调用云闪付开放平台接口置换授权令牌(auth_token)和用户标识(openId)；
5. 商家服务端后台使用授权令牌(auth_token)和用户标识(openId)调用云闪付开放平台接口获取授权用户信息。

2. 准备工作

APP 云闪付授权联登产品可以向云闪付合作商家 APP 输出云闪付用户信息产品，签署合作协议后可以申请集成云闪付快登产品软件包，以适应合作商家业务需求。

云闪付应用登录基于 OAUTH2.0 授权机制实现，提供登入，信息，扣款三种授权级别，商家 APP 集成云闪付 sdk 后通过 auth_token 可实现快登获取用户信息等功能（支付功能后续开通）。

合作商家 APP 提供以下接入方参数给云闪付开放平台配置：

接入方：[公司全称]	xxxx 公司
接入方简称：[公司简称]	xxxx
调用开放平台的出口 IP 和端口	xxx.xxx.xxx.xxx:端口
RSA 密钥文件	xxx _pub.pem
授权联登 logo	jpg/png
测试入口 URL	https://xxxx.net/login
联系方式和邮箱	phone/email

注：

1. 以上域名仅支持 https。
2. xxx _pub.pem 是接入方提供的公钥。

云闪付开放平台审核配置完成后，邮件返回合作商家 APP 以下信息：

appId： 合作商家 APP 唯一标识

secret： 开放平台分配给合作商家密钥，做签名因子

symmetricKey：云闪付提供的对称密钥（3DES, 16 进制格式），用于后台敏感数据解密

upPublicKey： 由云闪付提供，云闪付开放平台采用私钥对部分接口返回信息进行 RSA 签名， 合作接入方使用公钥 upPublicKey 验证签名；openssl 生成，base64 形式输出。(非必须)

3. 集成和调用云闪付 SDK

商家 app 需要集成配置云闪付提供的客户端 sdk，正确集成接入并拥护相关授权域 scope 后，通过 sdk 可唤起云闪付应用或打开 H5 进行授权登录，用户授权完成可拉起商户 app 应用，并带上授权码 code。
iOS 和 Android 集成配置和调用过程：
ios 参见《中国银联云闪付 iOS SDK 接入规范.pdf》;
Andriod 参见《中国银联云闪付 Android SDK 接入规范.pdf》;

4. 授权后台调用接口

1. 获取基础服务令牌 backendToken:
商家 app 后台调用接口获取基础服务令牌 backendToken，请求 url：
https://open.95516.com/open/access/1.0/backendToken
请求参数说明：

参数	是否必须	说明
appId	是	接入方的唯一标识
nonceStr	是	生成签名的随机字符串，由接入方自行生成，算法参照 FAQ
timestamp	是	生成签名的时间戳
signature	是	签名值，签名因子包括(appId, secret, nonceStr, timestamp)

返回参数说明:

参数	说明
backendToken	OAUTH2 后台接口调用凭证
expiresIn	backendToken 接口调用凭证超时时间，单位（秒）

说明：
(a) 签名算法，nonceStr 如何生成请参考 FAQ。
(b) timestamp 生成签名的时间戳，接入方生成从 1970 年 1 月 1 日 00:00:00 至今的秒数，即当前的时间，单位为秒。
(c) 以下所有接口签名都可参考以上两点。
(d) secret 为给接入方分配的密钥。
(e) backendToken 的有效期 expiresIn 现阶段返回 7200 秒，请做好缓存，后频繁调用云闪

付会将其列为黑名单拒绝访问，暂时没有限制。

2. 通过 code 获取 accessToken 和 openId:

合作商家 app 后台收到授权码 code 后请求云闪付开放平台后台 URL：

https://open.95516.com/open/access/1.0/token

请求参数说明：

参数	是否必须	说明
appId	是	商家 app 唯一标识
backendToken	是	基础服务令牌
code	是	授权码 code
grantType	是	直接填写常量字符串 authorization_code

返回参数说明：

参数	描述
accessToken	网页授权接口调用凭证
expiresIn	accessToken 接口调用凭证超时时间，单位（秒）
refreshToken	用户刷新 accessToken
openId	用户唯一标识
Scope	用户授权的作用域

返回示例：

```
{
 “accessToken”:”ACCESSTOKEN”,
 “expiresIn”:”3600”,
 “refreshToken”:”REFRESHTOKEN”,
 “openId”:”OPENID”,
 “scope”:”SCOPE”
}
```

backendToken 的有效期 expiresIn 现阶段返回 7200 秒，请做好缓存，后频繁调用云闪付会将其列为黑名单拒绝访问，暂时没有限制。

3. 获取用户登录信息：

商户 app 服务端获取到 accessToken 和 openId，调用开放平台接口可获取用户登录信息(即云闪付登录手机号)，请求 url:

https://open.95516.com/open/access/1.0/user.mobile

请求参数说明：

参数	是否必须	说明
appId	是	接入方的唯一标识
accessToken	是	OAUTH2 中获取的 accessToken
openId	是	OAUTH2 中获取的 openId
backendToken	是	4.1 获取到的 backendToken

返回参数说明：

参数	说明
mobile	登录手机号

说明：

1. 授权 scope 为 upapi_user 或 upapi_pay 才能访问此接口。
2. 手机号加密返回，请用云闪付分配给合作商家的对称秘钥（symmetricKey）解密，加密内容为 base64 格式输出。

4. 获取用户敏感信息+身份证号+姓名：

请求 url:

https://open.95516.com/open/access/1.0/user.auth

请求参数说明：

参数	是否必须	说明
appId	是	接入方的唯一标识
accessToken	是	OAUTH2 中获取的 accessToken
openId	是	OAUTH2 获取的 openId
backendToken	是	4.1 获取的 backendToken

返回参数说明：

参数	说明
realName	用户姓名
certTp	证件类型 （01：身份证，03：护照，04：回乡证，05：台胞证）
certId	证件号

说明：
1. 授权 scope 为 upapi_user 或 upapi_pay 才能访问此接口；
2. 此接口所有字段加密返回，请用云闪付分配给合作商家的对称秘钥（symmetricKey）解密，加密内容为 base64 格式输出。

5. FAQ

1. 云闪付开放平台后台请求报文格式如何定义？
回答：
所有后台请求均为 POST，JSON 报文格式，应答报文 JSON 格式如下：
resp: ‘00’表示成功，其他的都表示错误。
msg: 成功或者失败详细描述信息。
params: 具体数据返回，JSON 格式。

2. SHA256 签名算法如何实现？
将所有待签名参数按照字段名的 ASCII 码从小到大排序（字典序）后，使用 URL 键值对的格式(即 key1=value1&key2=value2…)拼接成字符串 string1。Key,Value 均采用原始值，大小写不变,不进行 URL 转义。 最后对拼接字符串 string1 作 sha256 运算出 signature。
signature=sha256(string1)
以 upsdk 签名为例；
- appId=a5949221470c4059b9b0b45a90c81527
- nonceStr=Wm3WZYTPz0wzccnW
- timestamp=1414587457
- secret= 388f9cb4a0df474883a32bec19da747f

步骤 1. 对所有待签名参数按照字段名的 ASCII 码从小到大排序（字典序）后，使用 URL 键值对的格式（即 key1=value1&key2=value2…）拼接成字符串 string1：
appId=a5949221470c4059b9b0b45a90c81527&nonceStr=Wm3WZYTPz0wzccnW&secret=388f9cb4a0df474883a32bec19da747f&timestamp=1414587457
步骤 2. 对 string1 进行 sha256 签名，得到 signature：
dadff952a1040fbdfd503f94127f26b8e31c9d1dfd5b8f9b175c64b26541aa8a

签名用的 nonceStr 和 timestamp 必须与请求参数中的 nonceStr 和 timestamp 相同。
SHA-256 签名参考：

```
public static String sha256(byte[] data) {
    try {
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        return bytesToHex(md.digest(data));
    } catch (Exception ex) {
        logger.info("Never happen.", ex);
        return null;
    }
}
/**
* 将 byte 数组转换成 16 进制字符串
*
* @param bytes
* @return 16 进制字符串
*/
public static String bytesToHex(byte[] bytes) {
    String hexArray = "0123456789abcdef";
    StringBuilder sb = new StringBuilder(bytes.length * 2);
    for (byte b : bytes) {
        int bi = b & 0xff;
        sb.append(hexArray.charAt(bi >> 4));
        sb.append(hexArray.charAt(bi & 0xf));
    }
    return sb.toString();
}
```

3. 如何生成签名用随机字符串 nonceStr?
请参考如下 PHP 实现:

```
private function createNonceStr($length = 16) {
    $str = null;
    $strPol = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz";
    $max = strlen($strPol)-1;
    for($i=0;$i<$length;$i++){
        $str.=$strPol[rand(0,$max)];//rand($min,$max)生成介于 min 和 max 两个数之间的一个随机整数
    }
    return $str;
}
```

4. 使用对称密钥解密方式：
symmetricKey 是银联分配给接入方的对称密钥，用于相关敏感信息的解密。加解密的方式为 3DES,并加密返回以 base64 编码输出，密钥以 16 进制的字符串形式提供。解密方法参考如下：

```
//用于将返回字段的值进行 3DES 解密
public static String getDecryptedValue(String value, String key) throws Exception {
         if (null == value || "".equals(value)) {
              return "";
         }
         byte[] valueByte = Base64.decode(value);
         byte[] sl = decrypt3DES(valueByte, BytesUtil.hexToBytes(key));
         String result = new String(sl);
         return result;
}

public static byte[] decrypt3DES(byte[] input, byte[] key) throws Exception {
         Cipher c = Cipher.getInstance("DESede/ECB/PKCS5Padding");
         c.init(Cipher.DECRYPT_MODE, new SecretKeySpec(key, "DESede"));
         return c.doFinal(input);
}
```