**用户SDK（IOS）嵌入文档**

更新时间: 2015/10/13 版本号1.1.0

目录

[**1.** **修订记录** 3](#_Toc422246612)

[**2.** **术语与缩写解释** 4](#_Toc422246613)

[**3.** **客户端接入方式** 5](#_Toc422246614)

[**3.1** **SDK文件清单** 5](#_Toc422246615)

[**3.2** **配置Xcode工程** 5](#_Toc422246616)

[**4.** **调用登录方法** 6](#_Toc422246617)

[**5.** **API文档** 7](#_Toc422246618)

[shareInstance 7](#_Toc422246619)

[startWithClientId 7](#_Toc422246620)

[**6.服务端接入方式** 8](#_Toc422246621)

[**6.1数字签名(重要)** 8](#_Toc422246622)

[**6.1.1签名策略** 8](#_Toc422246623)

[**6.1.2签名算法** 8](#_Toc422246624)

[**6.2登陆流程** 11](#_Toc422246625)

[**6.3服务端接口API(接口清单)** 12](#_Toc422246626)

[**6.3.1授权码兑换令牌** 12](#_Toc422246629)

[**6.3.2访问令牌鉴权** 14](#_Toc422246630)

[**附录Ⅰ：登陆错误代码** 16](#_Toc422246631)

[**服务端错误代码** 16](#_Toc422246632)

[**签名级别代码** 16](#_Toc422246633)

[**客户端错误代码** 16](#_Toc422246634)

[**附录II：Q&A** 17](#_Toc422246635)

1. **修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 作者 | 更新时间 | 更新内容 |
| 1.0.0 | 刘旦 | 2015.6.16 | 文档初始化 |
| 1.1.0 | 刘旦 | 2015.10.12 | 1.1版本的改动 增加对IOS9的适配 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**请特别注意文档中红色字体的描述部分！！！**

1. **术语与缩写解释**

|  |  |  |
| --- | --- | --- |
| 术语 | 定义 | 备注 |
| ClientId/client\_id | 应用编号 |  |
| ClientSecret/client\_secret | 应用密码 | 为32为md5码，用于应用和爱游戏服务器端通信时，对通信内容进行数字签名 |
| CP | 应用提供商 |  |
| token | 用户给应用的授权令牌信息 | 包含授权令牌，刷新令牌，有效期，是否失效状态，用户编号，应用编号等信息，爱游戏服务器颁发的令牌有效期为两个月 |
| access\_token | 授权令牌 | 为32位md5码 |
| code | 授权码，用于兑换授权令牌 | 为32位md5码，有效期十分钟 |

1. **客户端接入方式**
   1. **SDK文件清单**

在xCode的工程目录中新建如图3-1的文件夹EgameUserSdk, 并在xcode工程中通过Add Files to“工程名” 导入sdk所需文件。



图3-1 用户SDK文件列表

* 1. **配置Xcode工程**

**第一步: 加入系统Libraries**

MobileCoreServices.framwork

UIKit.framework

SystemConfiguration.framework

CFNetwork.framework

AdSupport.framework

libz.dylib

**第二步: 配置URL Schemes：**

打开Target的Info选项卡，找到URL Types并添加一个新项，Identifier为EgameUser，URL Schemes为自定义，建议为产品英文名字+EgameUser(不支持下划线等特殊字符)，如图3-4所示：



图3-4

1. **调用登录方法**

调用登录代码主要分为以下步骤：

1. **导入头文件：在需要调用登录代码的文件中导入以下头文件**

#import “EgameUserSdk.h”

1. **设置游戏参数：在需要调用登录代码的文件中设置游戏参数**

必要参数为游戏的Client\_id

//游戏的Client\_Id

#define CLIENT\_ID @"30681631"

获取SDK的单例并调用SDK的startWithClientId方法并处理登录结果

[[[EgameUserSdk shareInstance] startWithClientId:CLIENT\_ID callback:^(EgameResult result, NSString \*ssoCode, int failedReason) {

// 处理结果

NSLog(@"The sso code: %@", ssoCode);

NSLog(@"The result: %d", result);

NSLog(@"The failed reason: %d", failedReason);

}];

先判断EgameResult的result值。

SUCCESS: 登录成功，ssoCode为授权码，CP可以用来换取token。

FAILED: 登录失败，failedReason为错误码，可在附录I中查询。

CANCEL: 登陆取消，用户取消登陆。

1. **登录成功后根据授权码在有效时间内请自行换取token**
2. **API文档**

|  |  |
| --- | --- |
| **方法摘要** | |
| + (id) | **shareInstance**  单例模式，返回唯一实例 |
| - (void) | **startWithClientId:**(NSString \*) clientId  payCallback:(EgameCallbackBlock) callback  调用登录函数，在游戏登录之前执行 |
|  |  |

|  |
| --- |
| **方法详细信息** |

### shareInstance

+ (id) shareInstance

返回登录对象的单例

### startWithClientId

- (void) startWithClientId:(NSString \*) clientId

payCallback:(EgameCallbackBlock) callback

调用登录函数，在游戏登录之前执行

**参数：**

(NSString\*) clientId – 游戏的Client\_id

(EgamePayCallbackBlock) callback - 登录的回调函数

**6.服务端接入方式**

**6.1数字签名(重要)**

**6.1.1签名策略**

爱游戏开放平台的数字签名级别分三种，安全级别由低到高依次为：**无签名；基本签名；业务签名**。每个接口都会根据数据的安全要求，对数字签名级别进行限制，在服务请求过程中，如果签名的级别与要求的不一致，或者签名信息与请求参数不匹配，则会被认定为非法请求并阻止调用。所以请您在开发过程中，确保每个接口的签名都与接口所要求的级别、算法一致。

另外，我们为您提供了用于服务器交互的jar包及demo，包含发起http/https请求、数字签名等功能，如果您的服务器采用JAVA语言开发，可以直接引用，无需开发，如果您的服务器采用其他语言，请参考文档和demo自主实现相关功能。

**6.1.2签名算法**

**基本签名**

基本签名只对固定的基本参数进行签名，是CP身份的证明。

必须进行签名的字段：统一为client\_id、sign\_method、version、timestamp、client\_secret

具体算法如下：

在调用接口时，如果需要使用基本签名，需要在https请求参数中加入如下参数, 且参数名必须保持一致：client\_id、sign\_method、version、timestamp、sign\_sort、signature。

**client\_id:** 应用向开放平台注册时分配的应用编号

**sign\_method:** 数字签名的编码算法，如MD5，HmacSha1，HmacMD5等，SDK目前版本暂时只支持MD5算法

**version:** 所使用的SDK版本号，例如v2.1.0

**timestamp:** 请求发起时的时间戳，如：1385345938378

**sign\_sort：**用于指定请求参数在进行数字签名时的排列顺序。把client\_id、sign\_method、version、timestamp、client\_secret的参数名，按照任意顺序排列，然后拼接起来，参数名之间以“&”符号连接。例如：sign\_sort=client\_id&sign\_method&version&timestamp&client\_secret。另外请在签名后，请求发送前对此字段进行urlencode

**signature：**数字签名。计算步骤如下：

1. 把client\_id、sign\_method、version、timestamp、client\_secret参数的值，按照sign\_sort中指定的顺序排列，并拼接成字符串，参数之间无需分隔符。
2. 采用jdk的类java.security.MessageDigest，对步骤1生成的字符串进行md5计算，生成byte数组。
3. 将步骤2生成的byte数组转换成16进制字符串，此字符串即为数字签名。计算过程中字符集采用utf-8，16进制字符串转换算法必须与爱游戏提供的ByteFormat.java中算法保持一致。

**示例**：某应用（client=1001；client\_secret=a1b2c3），请求某接口

业务参数：token= aaaaaaaa,

签名级别：基本签名

则加入签名信息后最终请求参数如下:

token=aaaaaaaa

client\_id=1001

sign\_method=MD5

version=v2.1.0

timestamp= 1423018253631

sign\_sort= timestamp&sign\_method&client\_secret&client\_id&version

signature= bytesToHexString (MD5.encode(“1001v2.1.0MD5a1b2c31385345938378”))

**业务签名**

业务签名需要对固定的基本参数和接口业务参数进行签名。不仅能验证CP身份，而且能够提高数据安全性，防止在请求过程中业务参数被第三方篡改。所以采用业务签名的接口，会在接口清单中明确约定需要签名的字段。

必须进行签名的字段：具体接口指定。

具体算法如下：

在调用接口时，如果需要使用业务签名，需要在https请求参数中加入如下参数, 且参数名必须保持一致：client\_id、sign\_method、version、timestamp、sign\_sort、signature

**client\_id:** 应用向开放平台注册时分配的应用编号

**sign\_method:** 数字签名的编码算法，如MD5，HmacSha1，HmacMD5等，SDK目前版本暂时只支持MD5算法

**version:** 所使用的SDK版本号，如v2.1.0

**timestamp:** 请求发起时的时间戳，如：1385345938378

**sign\_sort：**用于指定请求参数在进行数字签名时的排列顺序。把接口要求进行签名的参数，按照任意顺序排列，然后拼接起来，参数名之间以“&”符号连接。例如：某接口约定使用业务签名，签名字段为client\_id、sign\_method、version、timestamp、client\_secret、username、password，则可以是sign\_sort=password&username&client\_secret&timestamp&version&sign\_method&client\_id。另外请在签名后，请求发送前对此字段进行urlencode

**signature**：数字签名。计算步骤如下：

1. 把所有需要进行签名的参数的值，按照sign\_sort中指定的排列，并拼接成字符串，参数之间无需分隔符。
2. 采用jdk的类java.security.MessageDigest，对步骤1生成的字符串进行md5计算，生成byte数组。
3. 将步骤2生成的byte数组转换成16进制字符串，此字符串即为数字签名。计算过程中字符集采用utf-8，16进制字符串转换算法必须与爱游戏提供的ByteFormat.java中算法保持一致。

**示例**：某应用（client=12；client\_secret=cs），请求某接口

业务参数：username=open； password=123； imsi=189；

签名级别：业务签名

签名字段：client\_id、sign\_method、version、timestamp、client\_secret、username、password、imsi

则加入签名信息后最终请求参数如下:

username=open

password=123

imsi=189

client\_id=12

sign\_method=MD5

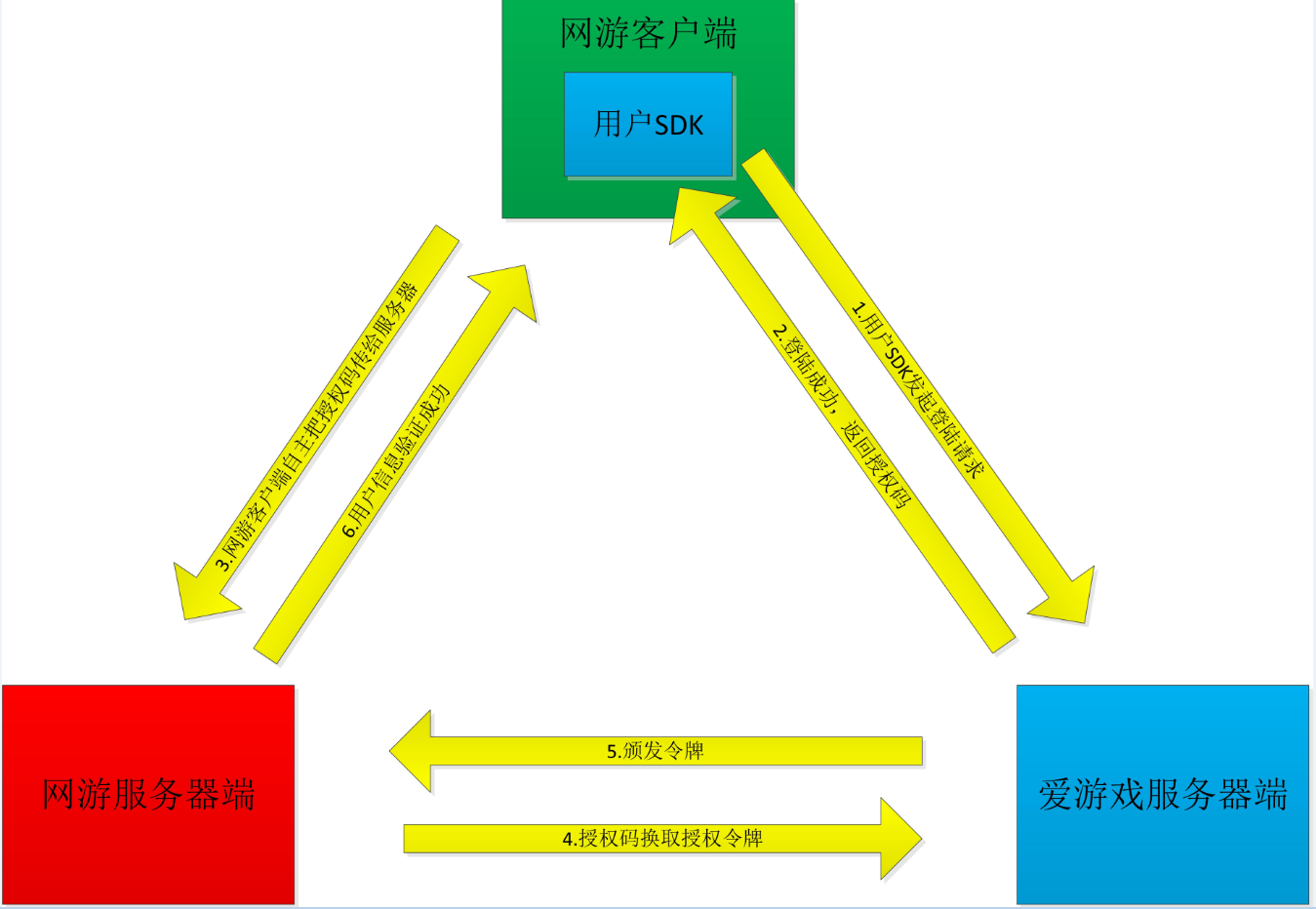
version=v2.1.0

timestamp=1385345938378

sign\_sort=client\_id&sign\_method&version&timestamp&client\_secret&username&password&imsi

signature= bytesToHexString (MD5.encode(“12MD5v2.1.01385345938378csopen123189”))

**6.2登陆流程**



登陆流程：

1. 用户触发登陆功能, 用户SDK判断用户是否满足自动登陆条件，如果不满足，引导用户进入登陆界面。
2. 用户身份通过验证后,爱游戏服务端会生成待登陆应用的授权码,并把授权码返回给客户端，提示登陆成功。
3. 网游客户端需要自主把授权码传给网游服务器，用于网游服务器换取授权令牌，确认用户信息。
4. CP服务器端调用爱游戏服务器端“授权码兑换token”接口（详见4.3.1）
5. 爱游戏服务器端校验授权码通过后，生成授权令牌并返回给CP服务器端。
6. CP服务器端收到令牌后，确认用户信息，之后便可以进行自己的业务逻辑（如绑定账号等）

**6.3服务端接口API(接口清单)**



**6.3.1授权码兑换令牌**

**服务地址：https://open.play.cn/oauth/token**

**签名级别：2**

**签名策略：基本签名**

**提交方式：POST**

**功能说明：获取访令牌**

**请求头参数：**Content-Type=application/x-www-form-urlencoded

**请求体参数：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **参数名** | **必填** | **类型/长度** | **备注** |
| **1** | client\_id | yes | String | 对应于应用注册时分配的client\_id |
| **2** | client\_secret | yes | String | 对应于应用注册时分配的client\_secret |
| **3** | code | yes | String | 用于兑换令牌的授权码 |
| **4** | grant\_type | yes | String | 此值固定为authorization\_code |
| **5** | scope | no | String | 访问请求的作用域 |
| **6** | state | no | String | 维护请求和响应的状态，传入值与返回值保持一致 |

**返回结果**

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **参数名** | **类型** | **备注** |
| **1** | access\_token | string | 访问令牌 |
| **2** | token\_type | string | 令牌类型，目前只有“Bearer” |
| **3** | refresh\_token | string | 刷新令牌 |
| **4** | expires\_in | string | 访问令牌过期时间 |
| **5** | re\_expires\_in | string | 刷新令牌过期时间 |
| **6** | scope | string | 授权范围 |
| **7** | user\_id | string | 爱游戏用户Id |

**可能的错误码：**

|  |  |
| --- | --- |
| 错误码 | 错误描述 |
| invalid\_request | 非法请求 |
| unauthorized\_client | 未授权的应用 |
| access\_denied | 请求被拒绝 |
| unsupported\_response\_type | 非法的response\_type |
| invalid\_scope | 非法的权限范围 |
| invalid\_grant | 非法的grant\_type类型 |
| expired\_token | 令牌过期 |
| insufficient\_scope | 超出授权范围 |
| scope limits | 不合法授权范围 |
| app status is invalid | 应用状态不可用 |
| appkey is overdue | appkey 超出有效时间 |
| app is overdue | 应用超出有效时间 |
| authorize\_code is overdue | authorize\_code超出有效时间 |
| access\_token is overdue | access\_token超出有效时间 |
| refresh\_token is overdue | refresh\_token超出有效时间 |
| request method must be get | 请求方式必须为get |
| request method must be post | 请求方式必须为post |
| client\_id is empty | client\_id 为空 |
| response\_type is empty | response\_type 为空 |
| redirect\_uri is empty | redirect\_uri为空 |
| grant type is empty | grant\_type为空 |
| authorize code is empty | 授权码为空 |
| client\_secret is invalidate | 应用的client\_secret无效 |
| app business exception occured | CP私有业务处理异常 |
| wrong version | SDK版本号为空或错误 |
| wrong sign\_sort | 签名顺序为空或错误 |
| wrong clientId | Client\_Id为空或错误 |
| wrong sign\_method | 签名算法为空或错误 |
| wrong timestamp | 时间戳为空或错误 |
| failed to check signature | 校验数字签名失败(爱游戏服务端错误) |
| wrong signature | 签名错误(与所传签名参数不匹配) |
| url\_write(get /oauth/token) is out of the config. | 请求方式错误，请使用post请求 |

**请求示例：**

https://open.play.cn/oauth/token?client\_id=111111&client\_secret=12341234473a415f9bd3a3e76609c15f&code=b57fe9b13125c13dc359864953759a95&grant\_type=authorization\_code&sign\_method=MD5&timestamp=1423019541343&sign\_sort=timestamp%26sign\_method%26client\_secret%26client\_id%26version&signature=173E82FEE2CFD50A63DA0AD58C78B910&version=2.1.0

**结果示例**

成功响应：

{

"scope": "all",

"re\_expires\_in": 15552000,

"user\_id": 956877,

"token\_type": "Bearer",

"expires\_in": 5184000,

"refresh\_token": "2c639e8c1cbfeee5fb07e968163d0343",

"access\_token": "2cd0a6f9c8ce81ada335f1989413ca08"

}

失败响应：

{

"error\_uri":null,

"error":"invalid\_request",

"state":null,

"error\_description":"Missing grant\_type parameter value"

}

**6.3.2访问令牌鉴权**

**服务地址：https://open.play.cn/oauth/token/validator**

**签名级别：2**

**签名策略：基本签名**

**提交方式：POST**

**功能说明：访问令牌鉴权（此接口主要用于鉴定用户身份，当应用客户端需要操作应用的数据时，如游戏道具，游戏金币等，应用服务器可以调用此接口鉴定用户身份，从而确定用户是否有权限操作相关数据）**

**请求体参数：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **参数名** | **必填** | **类型/长度** | **备注** |
| **1** | client\_id | yes | String | 对应于应用注册时返回的client\_id |
| **2** | access\_token | yes | String | 访问令牌 |

**返回结果：**

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **参数名** | **类型** | **备注** |
| **1** | code | string | 返回码,0表示成功 |
| **2** | text | string | 返回码描述信息 |
| **3** | ext | string | 业务数据 |
| **4** | ext->user\_id | string | 爱游戏用户编码 |
| **5** | ext->access\_token | string | 访问令牌 |
| **6** | ext->expires\_in | string | 令牌有效时间 |

**可能的错误码：**

-5,-260,-261,-262,-263,-264,-265

**请求示例：**

http://open.play.cn/oauth/token/validator?client\_id=111111&access\_token=3e8dee90ae7884cff0dfb1fa3818fa42&sign\_method=MD5&timestamp=1423019541343&sign\_sort=timestamp%26sign\_method%26client\_secret%26client\_id%26version&signature=173E82FEE2CFD50A63DA0AD58C78B910&version=2.1.0

**结果示例：**

成功响应：

{

"code": 0,

"ext": {

"access\_tpken": "6e87e2dd56dd92a9e5d0322ec9ecbaca",

"expires\_in": 4945807,

"user\_id": 317506

},

"text": "success"

}

失败响应：

{

"code":-261,

"text":"ErrorCode:-261 / Message:access\_Token is invalid",

"ext":null

}

**附录Ⅰ：登陆错误代码**

**服务端错误代码**

| 错误代码 | 错误信息 |
| --- | --- |
| -5 | 请求参数异常 |
| -260 | oauth 访问令牌accessToken过期 |
| -261 | oauth 访问令牌accessToken无效 |
| -262 | oauth 访问令牌accessToken失效 |
| -263 | 应用授权过期 |
| -264 | 应用访问受限 |
| -265 | 应用不存在 |
| -267 | 数字签名或签名参数错误 |

**签名级别代码**

| 签名级别代码 | 含义 |
| --- | --- |
| 1 | 无签名 |
| 2 | 基本签名 |
| 3 | 业务签名 |

**客户端错误代码**

| 错误代码 | 错误信息 |
| --- | --- |
| 0 | 无错误 |
| -10 | 登录失败 |
| -11 | 重定向失败 |
| -100 | 服务器异常 |
| -276 | 密码错误次数过多 |
| -1004 | 网络异常 |

**附录II：Q&A**

1. 移动和联通的手机号可以注册吗？

可以。

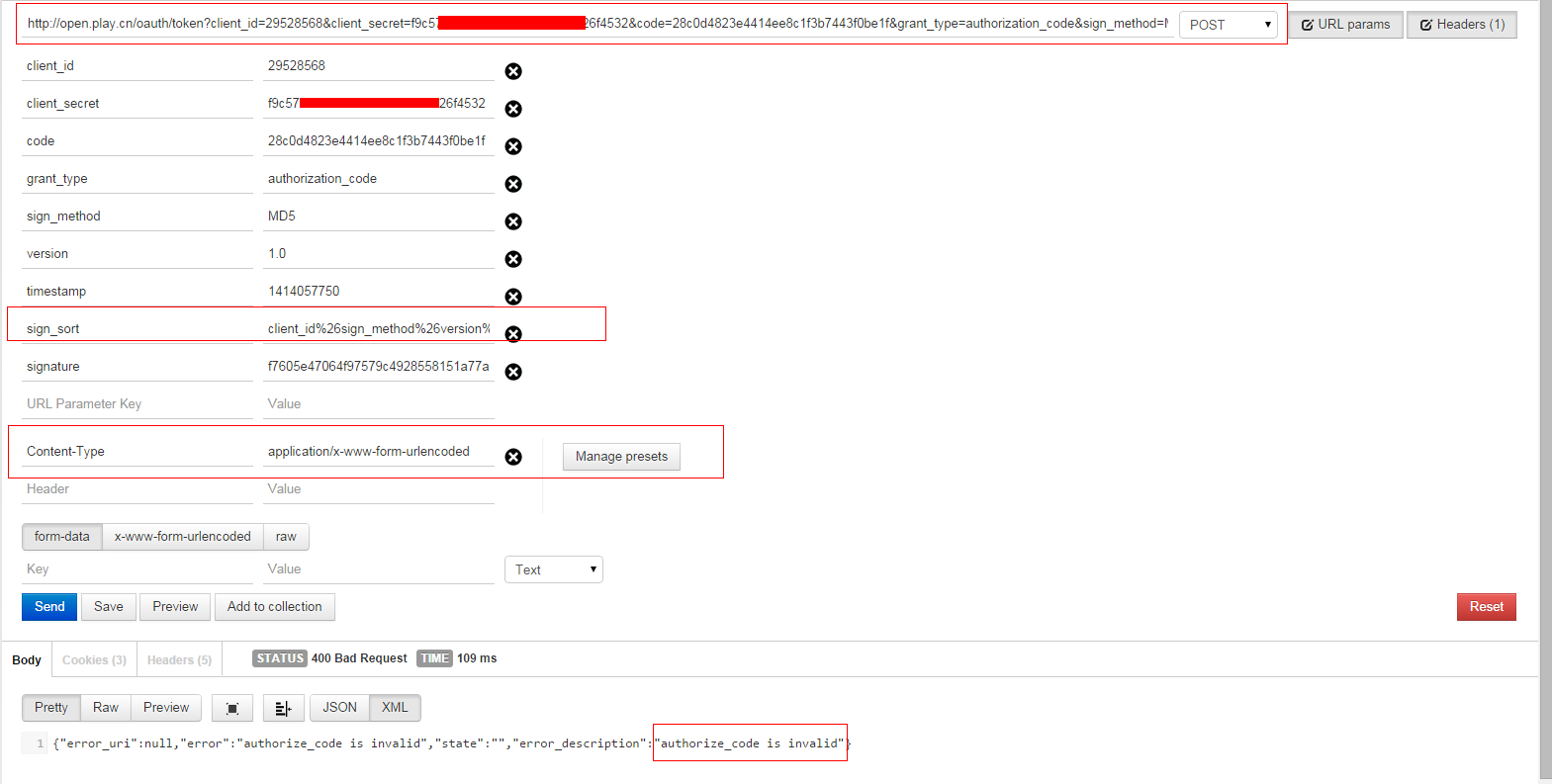
1. 为什么调用爱游戏授权码兑换令牌接口时，虽然做了签名，但是仍跑不通？

首先确定请求头里包含编码格式信息：

Content-Type = application/x-www-form-urlencoded

然后确定请求体里的参数已经做过UrlEcode处理，即参数值里包含的“&”等特殊字符已经转义。

如果按照以上方法仍无法跑通，请尝试把请求参数直接拼到url后面，而不是放到请求体里，如下图所示：



如果报错信息与图中一致，恭喜你，接口已经调通了，换一个正确的code即可。

1. 为什么做签名时，要把签名转换成16进制字符串？

因为MD5加密后的签名为byte数组，传输时需转换成对应字符串

1. 为什么我的网游服务端配置没有问题但是却拿不到客户端请求的数据？

因为虽然客户端重定向到网游服务端的提交方式是post，但是请求参数是拼接在url地址上的

1. 如何验证我的数字签名MD5算法是否与爱游戏一致？

可用您的MD5算法对如下串进行加密：

1421986517277MD5abc2d0c88900417c88e4f6d678c6d04f111111112.1.0

正确结果：4066426956C959ECBC84597786428313

1. 升级iOS9.0 之后无法登陆？

因为iOS9.0中的网络安全ATS 要求，导致SDK中的部分接口失效，目前解决方案如下。

你可以在Info.plist 配置中改用下面的XML源码：

<key>NSAppTransportSecurity</key>

<dict>

<key>NSAllowsArbitraryLoads</key>

<true/>

</dict>

在 plist 文件里显示如下：

../../../Library/Containers/com.tencent.qq/Data/Library/Application%20Support/QQ/Users/540401451/QQ/Temp.db/EEE99F0E-D971-4A63-B8BB-6962AE5821B2.png