开发参考文档

版本号	修订人	修订日期	修订内容
V1	艾堂勇/孔涛涛	2017-10-26	新建
V2	艾堂勇	2018-01-31	增加批量查询接口
V3	张辉	2018-06-10	2.3. 用户 VIP 等级查询接口地址由 http 变为 https，域名也有改动

目录

1. 概要

本文档用于指导 CP，接入 OPPO 平台游戏定制化运营相关功能。主要接口包括用户 VIP 等级查询、礼包发放接口。OPPO 公钥、加密算法、签名算法请参考附录。附录章节提供的示例代码是基于 JDK8 的，如果是 JAVA 平台，那么可以直接使用；如果是非 JAVA 平台，请自行实现。文档中的接口是纯服务端接口，无需客户端调用。

2. 服务端接口说明

2.1. 礼包发放接口

1) 接口描述：

此接口用于给用户发送礼包，如果礼包不允许用户重复领取，CP 需做好 accountId+realmId+roleId+giftId 的唯一限制，防止调用方超时后重试导致重复发送。一个游戏下的所有礼包，必须使用相同的发放 URL。

2) 请求方：OPPO 服务端

3) 请求地址：CP 提供

4) 调用方式：HTTP POST

5) 性能指标：TPS>500，响应时间<200m

6) **http 请求头：application/json**

7) **字符集：UTF-8**

8) **请求内容：请求格式（json）：**

字段名称	字段说明	类型	必填	备注
t	请求时间戳	string	Y	
data	请求数据	json	Y	BASE64(AES(data,**appSeceret 前 16 位**))
sign	签名参数	string	Y	BASE64(RSA(data,OPPO 私钥))
请求数据(对应 data 为 json 格式)				
pkg	游戏包名	string	Y	
giftId	礼包 ID	string	Y	
accountId	游戏账号	string	Y	
realmId	区服 ID	string	N	
roleId	角色 ID	string	N	

请求示例:

```
```

9) **返回内容：**

字段名称	字段说明	类型	必填	备注
code	请求的唯一标识	string	Y	
msg	结果描述	string		

结果示例：

```
{
    "code":20000,
    "msg":"OK"
}
```

响应码说明（code）：

响应码	说明	错误原因
20000	成功	
20001	该用户已经领取礼包	
50000	服务器内部错误	
40001	请求参数不正确	
40002	解密失败	
40003	验签失败	

2.2. 用户 VIP 等级查询接口

1）接口描述：此接口提供用户账号 vip 等级查询，用于 CP 发送 VIP 礼包的等级校验。

2）请求方：CP 服务端

3）请求地址：https://iopen.game.oppomobile.com/sdkopen/v2/vip/level

4）调用方式：HTTP POST

5）性能指标：TPS>500，响应时间<200m

6）http 请求头：application/json

7）字符集：UTF-8

8）请求内容：格式（JSON）：

字段名称	字段说明	类型	必填	备注
t	请求时间戳	string	Y	
client	调用方信息	json		
data	请求数据	json	Y	BASE64(AES(请求数据明文,**appSeceret 前 16 位**))
调用方信息(对应 data,采用 json 格式)				
pkg	游戏包名	String	Y	
请求数据(对应 data,采用 json 格式)				
accountId	游戏账号	string	Y	

请求数据示例:

```
{
    "client": {
        "pkg": "com.xxx.nearme.gamecenter"
    },
    "data":
"oVXiKk5ZK0tprOoGygW8qMh+0n9hZEhedKkcbu0WrpX99u7Z29T+Rqa9UFmwer5a",
    "t": 1510306865134
}
```

请求代码示例:

```
String appSecret = "BCefcf9Ae7550111777bbd3E78971834";

// 调用方信息
JSONObject client = new JSONObject();
client.put("pkg", "com.xxx.nearme.gamecenter");

// 请求数据
JSONObject data = new JSONObject();
data.put("accountId", "g140664427874260231");

// 加密后的 data
String cipherData = AesUtil.encrypt(appSecret.substring(0, 16),
data.toJSONString());

// http 请求参数
JSONObject request = new JSONObject();
request.put("t", String.valueOf(System.currentTimeMillis()));
request.put("client", client.toJSONString());
request.put("data", cipherData);
```

9）响应内容（JSON 格式）：

字段名称	字段说明	类型	必填	备注	
code	请求的唯一标识	string	Y		
msg	结果描述	string			
data	响应数据	json	Y		
响应数据(对应 data)					
vipLevel	Vip 等级	int			
vipName	Vip 名称	string			

结果示例：

```
{
    "code": "20000",
    "msg": "success",
    "data": {
        "vipLevel": 2,
        "vipName": "绿珀二星"
    }
}
```

响应码说明（code）：

响应码	说明	错误原因
20000	成功	
50000	服务器内部错误	
40001	参数不正确	
40002	解密失败	

3. 附录

3.1. AES 加解密 demo

```
import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;

public class AesUtil {

    public static String encrypt(String key,String text) throws Exception{
        Cipher cipher = Cipher.getInstance("AES/CBC/NOPadding");
        int blockSize = cipher.getBlockSize();
        byte[] dataBytes  = text.getBytes("UTF-8");

        int plaintextLen = dataBytes.length;
```

```
        if(plaintextLen % blockSize != 0){
            plaintextLen = plaintextLen + (blockSize - plaintextLen % blockSize );
        }
        byte[] plaintext = new byte[plaintextLen];
        System.arraycopy(dataBytes, 0, plaintext, 0, dataBytes.length);
        SecretKeySpec keySpec = new SecretKeySpec(key.getBytes("UTF-8"),"AES");
        IvParameterSpec ivSpec = new IvParameterSpec(key.getBytes("UTF-8"));
        cipher.init(Cipher.ENCRYPT_MODE, keySpec,ivSpec);
        byte[] encryptBytes = cipher.doFinal(plaintext);
        return Base64.getEncoder().encodeToString(encryptBytes);
    }

    public static String decrypt(String key,String cipherText) throws Exception{
        byte[] baseDecryptBytes =  Base64.getDecoder().decode(cipherText);
        Cipher cipher = Cipher.getInstance("AES/CBC/NOPadding");
        SecretKeySpec keySpec = new SecretKeySpec(key.getBytes("UTF-8"),"AES");
        IvParameterSpec ivSpec = new IvParameterSpec(key.getBytes("UTF-8"));
        cipher.init(Cipher.DECRYPT_MODE, keySpec,ivSpec);
        byte[] org = cipher.doFinal(baseDecryptBytes);
        return new String(org,"UTF-8").trim();
    }

    public static void main(String[] args) throws Exception {
        String encryptResult = encrypt("1234567890123456","hello world !");
        System.out.println(encryptResult);
        String decryptResult = decrypt("1234567890123456",encryptResult);
        System.out.println(decryptResult);
    }

}
```

3.2. RSA 验签 demo

```
import java.util.Base64;
import java.security.KeyFactory;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.security.Signature;

public class RsaUtil {

    public static final String charset = "utf-8";

    public static String sign(String content, String privateKey) throws Exception {
        PKCS8EncodedKeySpec priPKCS8 = new PKCS8EncodedKeySpec(Base64.getDecoder().decode(privateKey));
        KeyFactory keyf = KeyFactory.getInstance("RSA");
        PrivateKey priKey = keyf.generatePrivate(priPKCS8);
        Signature signature = Signature.getInstance("SHA1WithRSA");
        signature.initSign(priKey);
        signature.update(content.getBytes(charset));

        byte[] signed = signature.sign();
        return Base64.getEncoder().encodeToString(signed);
    }

    public static boolean check(String content, String sign, String publicKey) throws Exception{
        KeyFactory keyFactory = KeyFactory.getInstance("RSA");
        byte[] encodedKey = Base64.getDecoder().decode(publicKey);
        PublicKey pubKey = keyFactory.generatePublic(new X509EncodedKeySpec(encodedKey));
        Signature signature = Signature.getInstance("SHA1WithRSA");
        signature.initVerify(pubKey);
        signature.update(content.getBytes(charset));
        boolean cr = signature.verify(Base64.getDecoder().decode(sign));
        return cr;
    }

}
```

3.3. RSA 公钥

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCmreYIkPwVovKR8rLHWlFVw
7YDfm9uQOJKL89Smt6ypXGVdrAKKl0wNYc3/jecAoPi2ylChfa2iRu5gunJyNmpWZzlCN
RIau55fxGW0XEu553IiprOZcaw5OuYGlf60ga8QT6qToP0/dpiL/ZbmNUO9kUhosIjEu22uF
gR+5cYyQIDAQAB