**阿里巴巴**

**数娱事业部**

**ALi TV SDK 极简版 集成文档**

目 录

1 文档说明 4

1.1 业务约定 4

1.2 通信约定 4

1.3 安全规范 4

1.4 技术示例 5

1.4.1 RSA密钥生成方式 5

1.4.2 RSA签名(sign)算法示例 6

1.4.3 服务端notify解密demo（java版） 7

1.4.4 服务端notify解密demo（php版） 12

2 SDK接口说明 15

2.1 SDK接入方法 15

2.2 SDK系统接口 16

2.2.1 初始化 16

2.2.2 注销接口 17

2.2.3 获取SDK版本号 18

2.2.4 Log开关 18

2.3 登录接口 18

2.3.1 授权查询 18

2.3.2 登录授权接口 19

2.3.3 获取用户信息 20

2.4 支付接口 21

2.4.1网络支付接口 21

2.4.2 单机支付接口 28

# 文档说明

## 1.1 业务约定

三方应用接入，只能使用阿里提供的账号体系进行登录和授权，不能提供独立账号注册功能。

## 1.2 通信约定

支付服务端支持两种通信模式，即http（https）模式和TOP模式。

Https通信方式：适合于支付成功后的数娱服务端回调

使用https通信方式采用RSA公私钥加解密。该方式cp需要首先在数娱TV开放平台（http://open.aliplay.com/）填写开发者的RSA公钥信息。 数娱使用CP填写的公钥进行加签，CP的服务端接到回调后，使用私钥进行解签；回调的原始内容为json格式，具体字段定义参照SDK文档2.4.1.3节。根据原始内容json串以及RSA公钥加密后的内容encryptedData会在http参数的data字段中供CP利用RSA私钥解密后使用

TOP通信方式：适合于CP主动调用支付服务的场景（如查询订单状态等）

通过TOP通信方式，该方式CP需使用数娱TV开放平台（http://open.aliplay.com/）入住时颁发的appkey和appsecret（该信息可以在阿里TV开放平台应用管理页面中查看）， TOP接口的调用请参考<http://open.taobao.com/doc2/detail.htm?spm=a219a.7629140.0.0.jlQptD&treeId=49&articleId=101617&docType=1>

## 1.3 安全规范

为了防止双方约定的接口(服务端)被滥用带来的不必要的安全风险，接入的合作方需要符合以下要求做好安全控制。如有疑问，可加入旺旺群621730614 咨询。

1 接口安全规范

* 接口以HTTP方式开放。
* 设计接口需要有身份认证，对来源授权，只允许授权的IP访问。
* 所有的接口调用都需要有日志记录。

2 密钥及appsecret的传递及保存

接入合作方所持有的敏感信息，不应该大范围邮件传递，应以开发人员人对人传递。私钥和appsecret需要妥善保存，避免泄露私钥引起的安全隐患。

## 1.4 技术示例

### 1.4.1 RSA密钥生成方式

RSA是一种可用于进行签名的非对称算法，即签名密钥（私钥）与验签名密钥（公钥）是不一样的，私钥用于签名，公钥用于验签名。使用这种算法签名在起到防数据篡改功能的同时，还可以起到防抵赖的作用，因为私钥只有签名者知道。 可以使用Openssl RSA密钥生成工具，生成一套RSA公私钥。

**RSA密钥生成命令:**

生成 RSA私钥

openssl genrsa -out rsa\_private\_key.pem 1024

生成 RSA公钥

openssl rsa -in rsa\_private\_key.pem -pubout -out rsa\_public\_key.pem

将RSA私钥转换成PKCS8格式

openssl pkcs8 -topk8 -inform PEM -in rsa\_private\_key.pem -outform PEM -nocrypt

该工具生成的密钥文件，里面有一些注释和回车换行,需要将其去掉后组成一串字符!

### 1.4.2 RSA签名(sign)算法示例

**java参考代码**

 /\*\*

 \* RSA签名算法

 \* **@param** privateKey

 \* **@param** content

 \* **@param** charset

 \* **@return**

 \* **@throws** Exception

 \*/

 **public** **static** String sign(String privateKey, String content, String charset) **throws** Exception {

 PKCS8EncodedKeySpec priPKCS8 = **new** PKCS8EncodedKeySpec( Base64.*decodeBase64*(privateKey) );

 KeyFactory keyf = KeyFactory.*getInstance*("RSA");

 PrivateKey priKey = keyf.generatePrivate(priPKCS8);

 java.security.Signature signature = java.security.Signature.*getInstance*("SHA1WithRSA");

 signature.initSign(priKey);

 signature.update(content.getBytes(charset) );

 **byte**[] signed = signature.sign();

 **return** Base64.*encodeBase64String*(signed);

 }

|  |
| --- |
|  |

**php参考代码**

|  |
| --- |
| $privkey = file\_get\_contents("rsa\_private\_key.pem"); //无需转PKCS8$res = openssl\_get\_privatekey($privkey);openssl\_sign($data, $sign, $res);$sign = base64\_encode($sign); |

### 1.4.3 服务端notify解密demo（java版）

package testRsa;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.InputStream;

import java.io.UnsupportedEncodingException;

import java.security.InvalidKeyException;

import java.security.KeyFactory;

import java.security.NoSuchAlgorithmException;

import java.security.PrivateKey;

import java.security.SignatureException;

import java.security.spec.InvalidKeySpecException;

import java.security.spec.PKCS8EncodedKeySpec;

import javax.crypto.Cipher;

import org.apache.commons.codec.binary.Base64;

/\*\*

 \*

 \* @author zhiyong.xiongzy

 \*

 \*/

public class testRSA {

 private static String prikey = "MIICdwIBADANBgkqhkiG9w0BAQEFAASCAmEwggJdAgEAAoGBAMhDP5oVLHPJpM0QDuAGPupAkF4N3bMOD252BIYncKd8ppYJP0sA1wxnJxrVJN6UOzgK2qEReHUCzda5Hj+GaR8dmYz3v4UMozNeXhx7iqI7EK4Myp/qjCV2ROgh2LaL1ycugacTVx6dSfFvIOcczyhs8D4BUiJlSMsTWPVCSKsPAgMBAAECgYEAkL5KUgvLvUSZIL3ignkBMD7KGbDb9HbNIIGcc3o+KlAl93tni8ZnqEdGNFap91YcFz2BF7mQ3sHQFMpn2nJDtzj0tPsCKtNLHmpOi0ctAxedhzip+rOT6obNK+POL6czMWGJOaW8zLefxftgZWkT+eZactn507xc+GiwNsodAMECQQD4OjjnMhQWaSv8kHI27pH92PTv4zFb1msB0H65Im2zQ+8JrlqEH/7AqqMIxkPjIMC/vJLpkZG4TYMCYXjd4rx7AkEAzoiLGwUsXDApbDJJxxgYV6y50H/oJm6ooIOHoCwrluJhlOD81oUQyrLGfgNL95v4avvuQezZ4kmkeWDPvQ35fQJACBM8zqieZqw9NhYs4QSZ0zw/m540eNxc1s5FkRhBoVdQa2w0nZ+81d+3Ng3dH4JtQs+Lp/WIXAqJfIZXPECoCQJAW8QBgrTuu9tOXFuPul1zW2lMel6KiKD8Xa1zUnCtwXG+h8bbsHkZN+btGMpgM5libC6Z80LIoKm14ZRpWQffhQJBAJyxWEnzZA9dvqEhj6I/BTPJgvzc/boGzZ2lKQlN8j57MDkZHryoWtoKjB3Hp5dvTasweI6WHtIm313TSLuoXrg=";

 public static void main(String[] args) {

 String content = "c/BOhNHjhvCXvOenIyNqmr0FI1M3cYt/BR/WEoMgoi37ZFvKe418xOlRU4AN/QyvY2H7g2osL9oLmArsw35jk43reMFUyemTLnZ4Xw/DXT5xxPzHy4NWLt2/K0mqA2e5PiShf8RX9EdLnoIYZPXKe3Ldnw0ha3RZtP4X2tBsHqU=";

 String jsonStr;

 try {

 //解密

 jsonStr = decrypt(content, prikey, "UTF-8");

 System.out.println(jsonStr);

 //sign签名

 System.out.println(sign(prikey, content, "UTF-8"));

 } catch (Exception e) {

 e.printStackTrace();

 }

 }

 /\*\*

 \* RSA签名算法

 \* @param privateKey

 \* @param content

 \* @param charset

 \* @return

 \* @throws Exception

 \*/

 public static String sign(String privateKey, String content, String charset) throws Exception {

 PKCS8EncodedKeySpec priPKCS8 = new PKCS8EncodedKeySpec( Base64.decodeBase64(privateKey) );

 KeyFactory keyf = KeyFactory.getInstance("RSA");

 PrivateKey priKey = keyf.generatePrivate(priPKCS8);

 java.security.Signature signature = java.security.Signature.getInstance("SHA1WithRSA");

 signature.initSign(priKey);

 signature.update(content.getBytes(charset) );

 byte[] signed = signature.sign();

 return Base64.encodeBase64String(signed);

 }

 public static String decrypt(String content, String private\_key, String input\_charset) throws Exception {

 PrivateKey prikey = getPrivateKey(private\_key);

 Cipher cipher = Cipher.getInstance("RSA");

 cipher.init(Cipher.DECRYPT\_MODE, prikey);

 InputStream ins = new ByteArrayInputStream(Base64.decodeBase64(content.getBytes()));

 ByteArrayOutputStream writer = new ByteArrayOutputStream();

 //rsa解密的字节大小最多是128，将需要解密的内容，按128位拆开解密

 byte[] buf = new byte[128];

 int bufl;

 while ((bufl = ins.read(buf)) != -1) {

 byte[] block = null;

 if (buf.length == bufl) {

 block = buf;

 } else {

 block = new byte[bufl];

 for (int i = 0; i < bufl; i++) {

 block[i] = buf[i];

 }

 }

 writer.write(cipher.doFinal(block));

 }

 return new String(writer.toByteArray(), input\_charset);

 }

 public static PrivateKey getPrivateKey(String key) throws Exception {

 byte[] keyBytes;

 keyBytes = Base64.decodeBase64(key.getBytes());

 // 使用PKCS8

 PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(keyBytes);

 KeyFactory keyFactory = KeyFactory.getInstance("RSA");

 return privateKey;

 }

}

 PrivateKey privateKey = keyFactory.generatePrivate(keySpec);

### 1.4.4 服务端notify解密demo（php版）

<?php

$privkey=<<<EOT

-----BEGIN PRIVATE KEY-----

MIICdwIBADANBgkqhkiG9w0BAQEFAASCAmEwggJdAgEAAoGBAN/dCMqrU4GCvJ1s

QwPxvCnYSsXsm0+wdIzPBajdRiI9rvgtxRDq10yHGwXs+RNOGxaglj0F+9kRQ7l5

BEkN/HgPg90uWT6bNp5Qf4K3qJWfY6FmWoNDKYyIYjDacAVheAC/Idg/0nKgTUKH

sowx8iXHnKBB3o7B7XGoKCjFKc2PAgMBAAECgYEAjIVAqYdVyav5HZABmQ46oMGm

Xgshiqom3uRslw0eowu5Mwv38aCYOO9t5RBVYOrcUh3HcNK6vgz6SAhQylK/MKk8

hynSBnawh5/XZUUI7Zqw5NFoLIHgdKxcNzGhp5a6eTGOyPVXBi9/+PczeD5Enf2n

2RSCJV2yyAzxAzdM+DECQQD4eVHsYzkkmMFaPnJPWcGuLpl7hz1Fte5n83SiZhY/

6WJsC0FpKiCmpxrGhl2yM3gp0kS4b7/uuulIEQyPOAWdAkEA5qTiHZL/+GJOtxwU

kvvVplEmOKGHEcUSoLgxnBYS35S3qgxH4X989coyHV6EMNUXd5hO1vz3RnIVRvrx

sYsuGwJAcV9x2OlBNoiYmKOrprcun1pV375KX9duS3ITQtJPzqOfq+rK0sBNG51d

oz3F3LyNVI/DmWAr9DLGpIJa97I5JQJBALcqTmAKM7m/6J9Gb5hi4sMYQvBKZwfX

p/l3Df7VENv6MPmckyEjq8tObgog234kXGSf0kytzHpnczpRY8wJqe0CQCNv7Th8

jnlWQ21UVBL1Lr3KcMM/0RJ0hiVGH46Gp93h9c2zsAgHQ01HLi1BeRcM1tZZNPIh

GqEuuIa/iXjzNNc=

-----END PRIVATE KEY-----

EOT;

$res = openssl\_get\_privatekey($privkey);

//$content = "ycoYI7SXtj9GWHHNS27jeF18x7B0JVFxyb0+X/k4fjHpe7Jvwk7TGdT5VF+NWjBKpPVfyv5NRUYMNrHsHSeQgd4AWO6y3e5HWGmhCa92YdAzriSqoU2fAud4zJQtu1lKQP+iM5sC21GhSHlhm+2q443kKjuUX2Fl6wv4+H2onXBmbw2KHbz+knQW9AhjVoamYhy2X0XoAK6KtcQk7frk9ASqgf8iob0kteFp9HN/86+QFLUiH7BQm+Dh3tkDTBzIm4huCN4+bi71FLYLXLUYVWxcSC24S16Z11ihm771dFa+4w2taeAlhOEZHcE1w8uBXZPMlxlkT66OhvQXZRf88g==";

$content = $\_POST['data'];

$content = base64\_decode($content);

$i = 0;

$result = "";

**while** ($head = substr($content, $i, 128)) {

 $i += 128;

 openssl\_private\_decrypt($head,$newsource,$res);

 $result .= $newsource;

}

$result = json\_decode($result, **true**);

var\_dump($result);

# SDK接口说明

## 2.1 SDK接入方法

1 在eclipse中导入demo，能看到2个工程：



* AlitvSDKLib包含了SDK用到的lib和res
* AlitvSDKDemo引用上述Lib工程。
1. 按照以下截图配置，把SDK引入您的工程
2. 在项目属性中配置



1. 在Java Build Path中配置



1. 配置manifest可合并

在您的工程根目录中的project.properties里加入一行代码： manifestmerger.enabled=true

## 2.2 SDK系统接口

### 2.2.1 初始化

**接口说明**：

初始化SDK入口，输入必要的app信息。初始化的回调是异步的。

**前置条件**：

无

**函数原型**：

public static void init(Context context, String appKey, String appSecret, IInitListener initListener, IAuthListener authListener)

**参数说明**：

context应用程序上下文；

appkey 在淘宝top平台上申请的应用唯一ID;

appsecret在淘宝top平台上申请的应用对应的秘钥;

initListener 初始化回调接口,初始化完成之前，调用任何SDK接口都是无效的,

authListener 接收有关登录授权的所有回调

**回调接口原型：**

public interface IInitListener {

 public void onInitFinish();

 public void onInitError(String errorMsg);

}

public interface IAuthListener {

 public void onAuthSucess(int from);

 public void onAuthError(int errCode, String errMsg);

 public void onAuthCancel();

 public void onLogout();

}

**请求样例：**

AliTvSdk.*init*(*mContext*, appkey, appSecret, **new** IInitListener() {

 @Override

 public void onInitFinish() {

 TestToast.*show*("init aliTvSdk ok. get auth = " + AliTvSdk.Account.*isAuth*());

 }

 @Override

 public void onInitError(String errorMsg) {

 TestToast.show(errorMsg);

 }

 }, new TestAccount());

### 2.2.2 注销接口

**接口说明**：

释放SDK， 在程序退出时必须调用释放。

**前置条件**：

无

**函数原型**：

public static void destroy()

**参数说明**：

无

**请求样例**：

AliTvSdk.destroy();

### 2.2.3 获取SDK版本号

**接口说明**：

获取当前SDK的版本号。返回值为以下格式 “ostv\_x.x.x.x”, 用于调试所用。

**前置条件：**

无

**函数原型**：

public static String getSdkVersion()

**参数说明**：

无

**请求样例**：

AliTvSdk.getSdkVersion();

### 2.2.4 Log开关

**接口说明**：

Log信息打印，仅用于调试所用。为了运行效率，请在正式发布前关闭log打印

**前置条件：**

无

**函数原型**：

public static void logSwitch(boolean open)

**参数说明**：

open: true/false —> 打开/关闭；

**请求样例**：

AliTvSdk.logSwitch(true);

## 2.3 登录接口

账号相关接口的demo在TestAccount.java里

### 授权查询

**接口说明**：

判断用户是否授权，该接口立即返回当前授权状态。

若未授权可调用登录授权接口。

**前置条件：**

无

**函数原型**：

public static boolean isAuth()

**参数说明**：

无

**请求样例**：AliTvSdk.Account.isAuth()

### 登录授权接口

**接口说明**：

该接口，将调出登陆或授权页面。若未登陆则弹出登陆框，若已登录未授权，则弹出授权框。如果已经授权，立即返回true。

此接口会调用到盒子的账号服务程序，根据网络状况可能会耗时0.2～1秒，或更长。如果不是要发起授权，而是仅仅判断和是否已经授权，请使用2.3.1的查询接口

**前置条件：**

无

**函数原型**：

public static boolean checkAuthAndLogin()

**参数说明**：

无

**请求样例**：

 public static void checkAuthAndLogin() {

 boolean isAuth = AliTvSdk.Account.checkAuthAndLogin();

 if (isAuth) {

 TestToast.show("用户已授权");

 } else {

 TestToast.show("用户未授权，请扫码登陆");

 }

 };

### 获取用户信息

**接口说明**：

获取登录的用户信息，异步回调。返回值包括用户NickName、用户头像、用户数娱ID、用户积分。

请注意 默认情况下 系统不返回 NickName和用户头像，用户登录后可直接进入应用；若需返回NickName和用户头像，请联系运营配置，同时用户登录时会显示账号授权页面，在用户确认授权后 应用可能获取NickName和用户头像，用户没有授权则无法获取该信息。

**前置条件：**

需要授权，若未授权则返回未授权错误码

**函数原型**：

public static void getUserInfo(IGetUserinfoListener infoListener)

**参数说明**：

infoListener 接收回调

public interface IGetUserinfoListener {

 public void onSuccess(BaodianUserInfo userInfo);

 public void onError(int errCode);

}

**请求样例**：

AliTvSdk.Account.getUserInfo(new IGetUserinfoListener() {

 @Override

 public void onSuccess(UserInfo userInfo) {

 Log.d(TAG, "userinfo nick:" + userInfo.getUserNick() + ", id:" + userInfo.getUserId()

 + " headUrl:"+userInfo.getAvatarUrl());

 TestToast.show("nick:" + userInfo.getUserNick() + ", id:" + userInfo.getUserId()

 + " headUrl:"+userInfo.getAvatarUrl());

 } @Override

 public void onError(int errCode) {

 LogUtils.d(TAG,

 "get userinfo error:" + errCode + " " + AliBaseError.getErrMsg(errCode));

 TestToast.show(AliBaseError.getErrMsg(errCode) + "=====");

 }

 });

## 支付接口

本章将介绍 网络App（有服务端）和单机App（无服务端）接入支付方法。

支付接口的demo在TestPayActivity.java里

### 2.4.1网络支付接口

#### 2.4.1.1 支付流程说明



消费流程：

1. 用户在App中发起支付，App调用SDK支付接口payFromServer。
2. SDK与Ali服务端进行支付确认后，向用户显示支付页面
3. 用户确认支付后，SDK向Ali服务端发起并完成支付
4. Ali服务端 将支付结果反馈给SDK，并通知App服务端
5. App服务端通过回调的订单信息，决定是否向App的用户发放道具

#### 网络应用单次支付接口

**接口说明**：

App服务端将支付参数传递给App，App调用支付接口payFromServer完成支付

阿里服务端将支付结果通知SDK和App服务端，App服务端决定道具是否发放

**前置条件**：

无

**函数原型**：

public static void payFromServer(String title, String amount, String orderId, String notifyUrl,String subjectId, IPayListener payListener)

**参数说明**：

title：支付物品名称 ，如“宝石”

 amount：人民币“分” （如amount =100，代表1元人民币）

 subjectId 商品ID（用于参与激励活动商品标示，由CP自定义）可选

 orderId：订单号，CP的服务端生成

 notifyUrl：支付结束后，阿里服务器回调给CP服务器的回调地址

payListener：回调函数，支付结束后通知App客户端成功或者失败。

**请求样例**：

 **String** amount = “2”;

 String title = "支付人民币 0.02元";

 **final** String orderId = System.*currentTimeMillis*() + "";

 // App服务端提供回调接口，用于接收支付成功的消息

 String notifyUrl = "http://xxxx.com/callback/notify";

 AliTvSdk.*payFromServer*(title, amount , orderId , notifyUrl, subjectId,

 **new** IPayListener() {

 @Override

 **public** **void** onSuccess(String title, **int** amount) {

 TestToast.*show*("二维码扫码支付成功");

 }

 @Override

 **public** **void** onError(String title, **int** amount, String errMsg) {

 TestToast.*show*("二维码扫码支付失败, 原因：" + errMsg);

 }

 @Override

 **public** **void** onCancel(String title, **int** amount) {

 TestToast.*show*("二维码扫码支付取消");

 }

 });

#### 网络应用续订支付接口

##### 2.4.1.3.1 续订服务首次支付

**接口说明**：

public static void repayFromServer，实现自动续费功能，应用若需接入续费支付接口，需联系阿里运营配置开通。首笔续费将通过客户端实时完成，后续续费通过应用的服务端调用数娱的服务端实现。请注意续费接口的规则，不满足规则的调用将无法实现首笔支付或后续支付。

**前置条件**：

无

**函数原型**：

**public static void repayFromServer** (String subject, String orderId, String interval, List<String> priceList, String description, **final** String title, **final** String notifyUrl, **final** String subjectId, **final** IPayListener payListener)

**参数说明**：

subject：续费商品包，如“影视包月”

orderId：订单号，CP的服务端生成

interval：自动续费模式 (month：按月、season：按季)，只支持两种固定模式；无论按月或按季度都仅支持1年内的续费，即按月续费时，后续应用服务端最多可发起11次续费请求，按季续费后续最多可发起3次续费。

priceList：价格列表，包含本次消费，后续每次续订消费金额

description（可选）：业务权益结束时间，若没有则为空

title：支付目标商品 ，如“xx影视”

notifyUrl：支付结束后，阿里服务器回调给CP服务器的回调地址

 subjectId（可选） 商品ID，如为空，则无法参加消费激励活动，建议必填。

payListener：回调函数，支付结束后通知成功或者失败。

**请求样例**：

 AliTvSdk.*pay*("XX商品", "12345678", "month",

 Arrays.*asList*(arr), "权益描述",

 "支付物品名称", "http://setbycp2", **null**,

**new** IPayListener(){

 @Override

 **public** **void** onSuccess(String title, **int** amount) {

 TestToast.*show*("支付成功");

 }

 @Override

 **public** **void** onError(String title, **int** amount, String errMsg) {

 TestToast.*show*("支付失败, 原因：" + errMsg);

 }

 @Override

 **public** **void** onError(String errCode, String errMsg) {

 TestToast.*show*(" 支付失败。" + errCode + ":" + errMsg);

 }

 @Override

 **public** **void** onCancel(String title, **int** amount) {

 TestToast.*show*("支付取消");

 }

});

##### 2.4.1.3.2 续订服务后续支付

**接口说明**：

商户在使用客户端成功创建续费序列后，可以通过服务端，向支付后台发送续费请求。

**接入方式：TOP**

TOP接入指南见

<http://open.taobao.com/doc2/detail.htm?spm=a219a.7629140.0.0.jlQptD&treeId=49&articleId=101617&docType=1>

TOP辅助工具类（java）见附录。

**API：**

taobao.tvpay.partner.renewal.pay

**请求参数结构：**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数说明** | **是否必填** | **详情** |
| partner\_notify\_url | 回调地址 | 是 |  |
| partner\_order\_no | 该笔订单的商户订单号 | 是 |  |
| price | 价格，以人民币分为单位 | 是 | 需和最初创建序列时一直，否则会报错。 |
| ref\_order\_no | 序列首笔订单商户订单号 | 是 | 从sdk创建序列时，该笔订单对应的商户订单号。 |
| Seq | 续费序列号。 | 是 | SDK上支付的首笔seq算作0，从后台续费的第一笔seq为1，类推。 |
| Subject | 商品标题 | 是 |  |
| subject\_id | 商品id | 是 |  |

**返回基础结构说明：**

|  |  |
| --- | --- |
| **字段** | **说明** |
| Code | success:成功 |
| message | 提示信息或错误信息 |
| data.data | 返回业务参数（需要RSA方式使用私钥解密，得到json串并解析成结构化的data参数） |

**返回基础结构样例：**

{

 "tvpay\_partner\_renewal\_pay\_response":{

 "result":{

 "code":"success",

 "data":{

 "order\_status":"TRADE\_SUCCESS"

 },

 "message":"success",

 "success":true

 }

 }

}

如果请求成功（success为true），参考order\_status字段校验订单状态

WAIT\_BUYER\_PAY 等待买家支付

TRADE\_SUCCESS 支付成功

TRADE\_CLOSED 交易关闭

如果请求失败，请参考错误码code字段，分析错误原因。

TERM\_COUNT\_EXCEED\_LIMIT，账期过多

RENEWAL\_PRICE\_ERROR，金额校验未通过

RENEWAL\_PRICE\_EXCEED\_LIMIT 续费总额超过限制

RENEWAL\_ORDER\_NOT\_ALLOWED 应用暂不支持续费

RENEWAL\_ORDER\_INFO\_ERROR 订单信息有误

RENEWAL\_THREAD\_NOT\_FOUND 未找到续费序列

RENEWAL\_THREAD\_STATUS\_ERROR 续费序列状态异常

RENEWAL\_ORDER\_PAY\_FAIL 续费订单支付失败

RENEWAL\_TIME\_ERROR 未到续费时间

RENEWAL\_THREAD\_SEQ\_ERROR 续费账期有误

#### 服务器端支付成功后回调通知

第三方应用需要开发一个HTTP服务，并提供的服务器接口地址，比如：http://第三方应用服务端地址/XXX

数娱服务端使用商户提供的公钥（public\_key）对参数加密生成加密串encryptString（解密出来为json格式），并在app请求提供的URL（参数为：notify\_url）中加入本次成功的订单信息加密串（格式为：notify\_url?data=encryptString）

数娱服务端将上述信息 向APP服务端发送http请求，若收到返回信息为“success”，则发送成功，若返回信息为其他，递增时间间隔重发。

**Data解密后数据结构：**

|  |  |
| --- | --- |
| **键名(key)** | **值(value)** |
| partner\_order\_no | 商户订单号 |
| alipay\_trade\_no | 支付宝交易流水号 |
| order\_status | 订单状态  |
| WAIT\_BUYER\_PAY 等待买家支付 |
| TRADE\_SUCCESS 支付成功 |
| TRADE\_CLOSED 交易关闭 |
| buyer\_logon\_id | 付款方支付宝账号 |

#### 服务器端主动查询订单支付状态

**接口说明**：

商户可以通过该接口主动查询订单支付状态，主要用于客户端支付成功后，商户接到收支付成功后主动查询一次，用于验证订单是否已支付成功。

**接入方式：TOP**

TOP接入指南见

<http://open.taobao.com/doc2/detail.htm?spm=a219a.7629140.0.0.jlQptD&treeId=49&articleId=101617&docType=1>

TOP辅助工具类（java）见附录。

**API：**

taobao.tvpay.partner.order.query

**请求参数结构：**

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **参数说明** | **是否必填** | **详情** |
| order\_no | 商户订单号 | 是 | 商户自己的订单号 |

**返回基础结构说明：**

|  |  |
| --- | --- |
| **字段** | **说明** |
| code | success:成功 |
| message | 提示信息或错误信息 |
| data.data | 返回业务参数（需要RSA方式使用私钥解密，得到json串并解析成结构化的data参数） |

**返回基础结构样例：**

{

 "tvpay\_partner\_order\_query\_response": {

 "result": {

 "code": "success",

 "data": {

 "data": "PG5Qgxp5C+Muqohd7s01IE9SK/VnYVSxsY7nPqHFR8d+tjx3qLBeTu8llVi0+yaQHdMl/ZMTXxhchl9FiruOkX+GL5aN4S+/C68Tq5Os5dT4iilwNaDT6gw2pBtMUR9maVfXukcvFMLSx7d9XsjebWpapqTMgjOkMQjOHUA0GtcdgfFPE4sFxwB0+8SdbOYF1qoEkBoU8o0brd6/tFTJkqn/uJUbHOoqqLXCB3LmxShRa4m8KRsVPfSavOEjcSO7yfNTu7cLhGoOCRL5ffwLWVxyPRpaLfm0O8CJi9Cllb5nndWzh1zoXUtyBdfw9iHzbE0C7WArO5+YLKVCzA5c4A=="

 },

 "message": "success",

 "success": true

 },

 "request\_id": "118g71ywn59zf"

 }

}

|  |
| --- |
|  |

**data解密后数据结构：**

|  |  |
| --- | --- |
| **键名(key)** | **值(value)** |
| partner\_order\_no | 商户订单号 |
| alipay\_trade\_no | 支付宝交易流水号 |
| order\_status | 订单状态  |
| WAIT\_BUYER\_PAY 等待买家支付 |
| TRADE\_SUCCESS 支付成功 |
| TRADE\_CLOSED 交易关闭 |
| price | 订单价格，以人民币分为单位 |
| subject | 商品标题 |
| subject\_id | 商品id |
| buyer\_logon\_id | 付款方支付宝账号 |

### 2.4.2 单机支付接口

#### 支付流程说明



消费流程：

1. 用户在App中发起支付，调用接口public static void pay
2. SDK与Ali服务端进行支付确认后，向用户显示支付页面
3. 用户确认支付后，SDK向Ali服务端发起并完成支付
4. Ali服务端 将支付结果反馈给SDK
5. SDK回调里返回支付状态
6. App根据回调状态，决定是否为App的用户发放道具

#### 单机应用支付接口

**接口说明**：

单机App调用支付接口 public static void pay

App客户端通过回调状态，决定道具是否发放

**前置条件**：

无

**函数原型**：

public static void pay(String title, String subjectId, String amount, IPayListener payListener)

**参数说明**：

title：支付物品名称 ，如“宝石”

 subjectId 商品ID（用于参与激励活动商品标示，由CP自定义）可选

amount: 人民币“分” （如amount =100，代表1元人民币）

payListener：回调函数，支付结束后通知成功或者失败。

**请求样例**：

 **String** amount = “1”;

 String title = "支付人民币 0.01元";

 AliTvSdk.*pay*(title, subjectId ,amount, **new** IPayListener() {

 @Override

 **public** **void** onSuccess(String title, **int** amount) {

 TestToast.*show*("支付成功");

 }

 @Override

 **public** **void** onError(String title, **int** amount, String errMsg) {

 TestToast.*show*("支付失败, 原因：" + errMsg);

 }

 @Override

 **public** **void** onCancel(String title, **int** amount) {

 TestToast.*show*("支付取消");

 }

});