同步游戏联运 SDK 接入参考手册

V1.4.0

版本	更新说明	更新时间
1.0.2	1、修改招商银行支付失败的问题 2、添加渠道信息 3、修改部分界面兼容性问题 4、修复升级失败的问题 5、优化升级模块 注：无接口修改	2015-02-18
1.0.3	1、修改部分机型初始化失败闪退的问题 2、修改悬浮窗兼容性问题 3、修改招商银行支付细节	2015-04-15
1.1.0	1、修改部分细节 2、优化版本信息	2015-05-08
1.2.0	1、添加第三方登录 2、添加手机注册 3、其他细节优化	2015-07-15
1.3.0	1、添加支付密码 2、添加微信支付 3、更新银联 sdk，支持 64 位系统 4、更新支付宝 sdk 5、其他细节和界面优化	2015-11-10
1.4.0	1、更新银联 sdk 2、其他细节和界面优化	2017-05-18

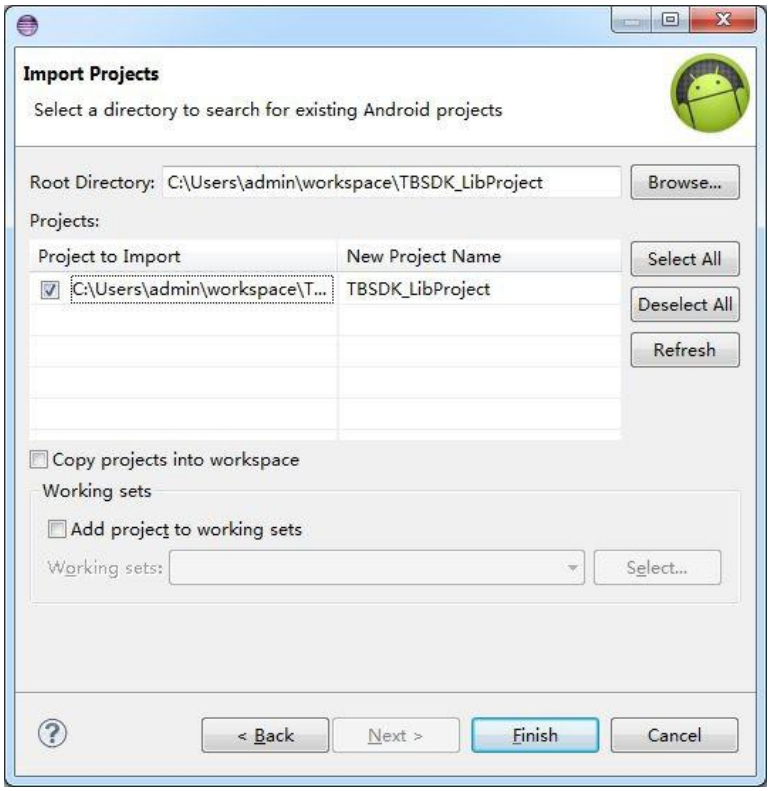
开发前需知

在进行开发前，请先注意以下事宜：

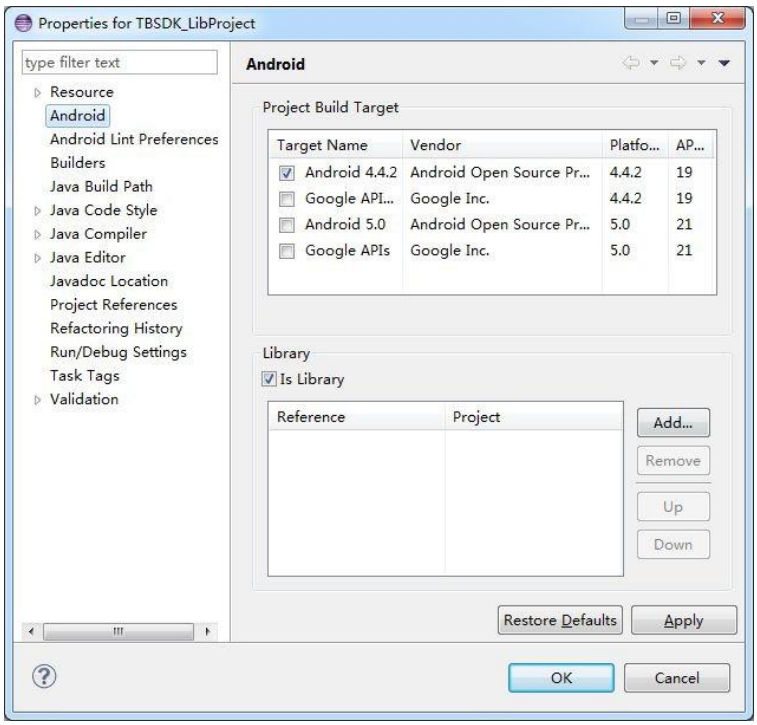
1. 确保双方商务已完成合作事宜的洽谈，并签订合作协议；
2. 检查您是否已在同步商务创建的讨论组中，如果没有，请尽快联系商务将您加入讨论组中，以及时获得帮助；
3. 登陆游戏联运中心 http://dev.tongbu.com/game/ 获取 Appid 和 Appkey。若 应用列表为空，请添加应用，并联系我方商务进行审核；
4. 支付通知地址、代币单位名称、代币兑换比例、个人中心是否显示切换账号按钮和是否强制升级，均可在游戏联运中心进行配置；
5. 开发中遇到任何问题或不清楚的地方，请在讨论组中提出。

一、SDK 构成 二、开发环境搭建

1. 导入 TBSDK_LibProject 工程 在您的工程所在工作空间下导入 TBSDK_LibProject 工程：



点击该工程，选择 Properties 进入界面后选择 Android 并将 Is Library 勾选，如图：



2. 为您的工程引入 TBSDK_LibProject

3. 当成功 import 进 TBSDK_LibProject 工程后您就可以为您的工程引入 TBSDK 了，步骤如下：

右键点击您的工程，选择 Properties 后选择 Android，在面板上点击 Add 将 TBSDK_LibProject 工程引进来。

4. 配置 AndroidManifest.xml

您需要将 AndroidManifest.xml 配置说明.txt 里面的权限、Activity、Service 等拷贝到您的工程的 AndroidManifest.xml 里面

权限：

```
<!-- permission -->
<uses-permission android:name="android.permission.GET_TASKS" />
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
```

渠道信息：

```
<meta-data
    android:name="TONGBU_CHANNEL"
    android:value="tongbu" />
```

Activity、Service 等：

```
<application android:allowBackup="true" >

    <!-- Activity -->
    <activity
        android:name="com.tongbu.sdk.ui.TbMainActivity"
        android:configChanges="orientation|keyboardHidden|navigation|screenSize"
        android:launchMode="singleTask" >
    </activity>
    <activity
        android:name="com.tongbu.sdk.ui.TbAccountLoginActivity"
        android:configChanges="orientation|keyboardHidden|navigation|screenSize" />
    <activity
        android:name="com.tongbu.sdk.ui.TbAccountRegisterActivity"
        android:configChanges="orientation|keyboardHidden|navigation|screenSize" />
    <activity
        android:name="com.tongbu.sdk.ui.TbCustomWebActivity"
        android:configChanges="orientation|keyboardHidden|navigation|screenSize" />
    <activity
        android:name="com.tongbu.sdk.ui.TbSdkCenterActivity"
        android:launchMode="singleTask" />

    <!-- alipay sdk begin -->
    <activity
        android:name="com.alipay.sdk.app.H5PayActivity"
        android:configChanges="orientation|keyboardHidden|navigation"
        android:exported="false"
        android:screenOrientation="behind"
        android:windowSoftInputMode="adjustResize|stateHidden" />
    <!-- Service -->
    <service android:name="com.tongbu.sdk.service.TbUpdateService" />
</application>
```

三、接入同步 SDK

1.平台初始化

为保证后续接入平台其他功能的正常工作，建议在游戏启动完毕后马上进行平台初始化工作（该方法会进行游戏版本检查等工作）。

函数：

```
void tbInit(Context mContext, TbPlatformSettings settings, OnInitFinishedListener listener)
```

Context : 开发者游戏启动页的上下文

TbPlatformSettings : appId 和 orient 等相关的参数设置

OnInitFinishedListener : 初始化完成后的通知,开发者应该在改通知中继续相关 工作（如登陆）。

初始结果回调接口如下：

```
interface OnInitFinishedListener {
    /**
     * @param flag 初始结果状态码
     */
    void onInitFinished(int flag);
}
```

调用结束后返回：

对应的状态码（OnInitFinishedListener 中定义）	描述
FLAG_NORMAL	正常初始化游戏
FLAG_FORCE_CLOSE	强制退出游戏
FLAG_CHECK_FAILED	检查更新失败

2.设置平台横竖屏方向（TbPlatformSettings 中设置）

相关参数

对应的状态码（TbPlatformSettings 中定义）	描述

SCREEN_ORIENTATION_PORTRAIT	竖屏（默认值）
SCREEN_ORIENTATION_LAND	横屏
SCREEN_ORIENTATION_AUTO	自动（由重力感应自动选择）

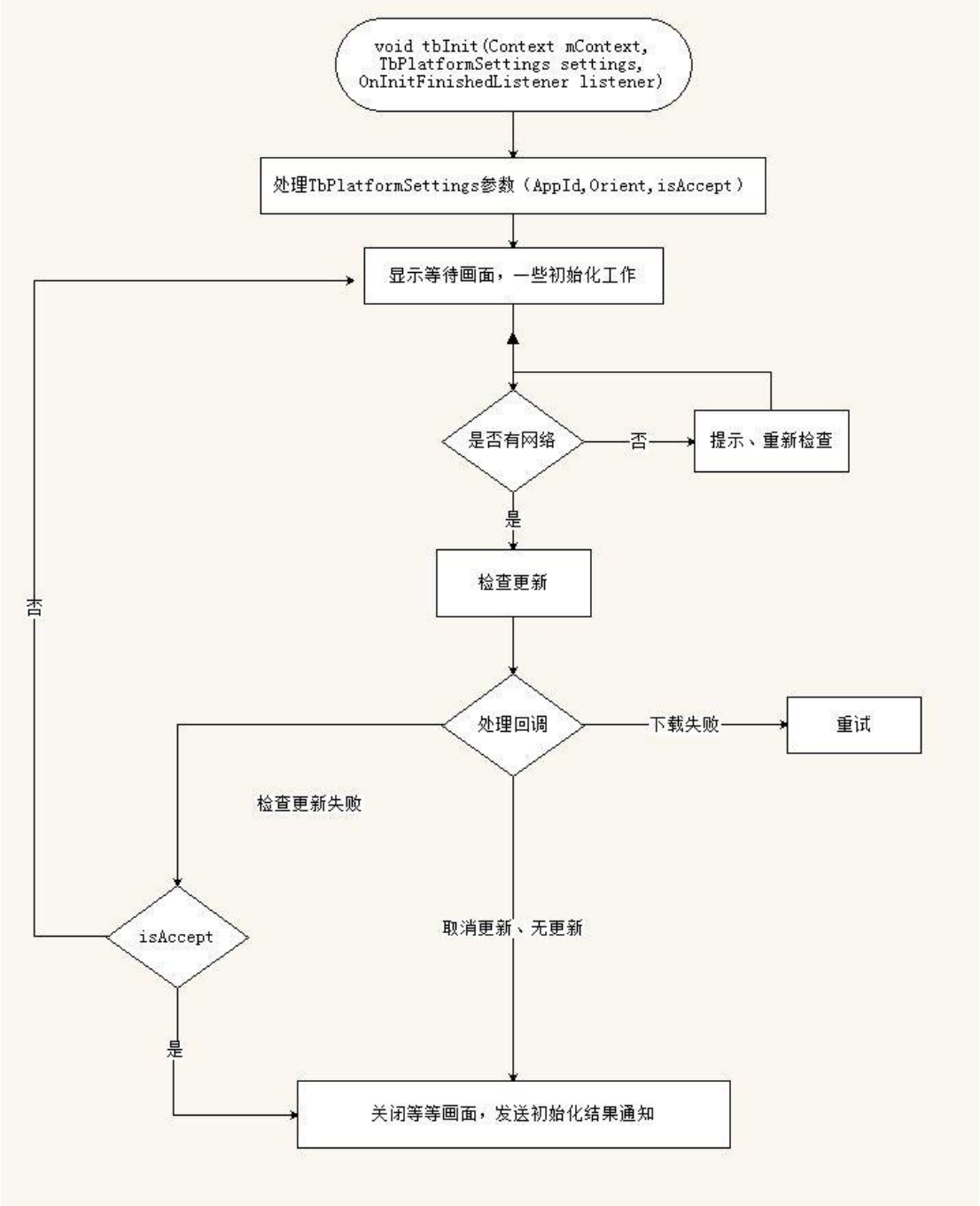
3.检查更新 平台初始化工作包含了游戏的检查更新工作，检查更新分为两种模式，分别为： 强制更新：玩家需要更新完才能继续游戏。 非强制更新：玩家不需要更新也可以进行游戏。

*是否强制更新请到联运后台中设置，如图：

注：确保游戏内配置的版本号与所计划发布的版本号一致，每次升级需要记得调整该处的版本号。

平台初始化接口将自动进行版本检查工作，开发者在设置初始化参数(TbPlatformSettings)时需要设置在检查失败时（网络错误或服务器错误）是否允许进入游戏(setContinueWhenCheckUpdateFailed())，如果允许进入游戏，则在初始化结果通知里需要对检查失败的情况做处理，否则可能导致强制更新可跳过等问题。

自动检测更新的大致流程图：



注：初始化参数中 isDebug()是测试更新流程使用的，设为 true 之后会强制弹出更新

4.释放平台内存

游戏退出时务必调用本接口，以释放 SDK 资源以及保存相关数据。

函数：

```
void tbDestory(Context mContext)
```

5. 显示/隐藏悬浮工具条

悬浮条包含个人中心、系统公告等入口。

1.创建悬浮工具条

函数：

```
TbFloatToolBar create(Context context, int place)
```

在要显示浮动工具栏的 Activty 的 onCreate 方法中调用该函数，并将浮动工具栏对象保存 为成员变量。

如：

```
private TbFloatToolBar toolBar;
@Override
protected void onCreate(Bundle savedInstanceState)
    { super.onCreate(savedInstanceState);
    toolBar = TbFloatToolBar.create(this, TbToolBarPlace.POSITION_BOTTOM_LEFT);
}
```

Place 位置设置：

对应的状态码（TbToolBarPlace 中定义）	描述
POSITION_TOP_LEFT	左上角
POSITION_TOP_RIGHT	右上角
POSITION_MID_LEFT	左边中间
POSITION_MID_RIGHT	右边中间
POSITION_BOTTOM_LEFT	左下角
POSITION_BOTTOM_RIGHT	右下角

2.显示悬浮工具条

函数：

```
void show()
```

在要显示浮动工具栏的 Activty 中的合适时机调用该函数，例如 onCreate() 方法中。 注意：调用该接口前请确保调用过创建浮动工具栏接口。

3.隐藏悬浮工具条

函数：

```
void hide()
```

4.回收悬浮工具条

函数：

```
void recycle()
```

在 Activity 销毁的时候调用。回收当前 Activty 中的浮动工具栏实例。

如：

```
@Override
    protected void onDestroy()
        { if (toolBar != null) {
            toolBar.recycle();
            toolBar = null;
        }
        super.onDestroy();
    }
```

6.平台登录、注销、切换账号

1.登录

函数：

```
void tbLogin(Context context, OnLoginFinishedListener listener)
```

说明：该接口在玩家有自动登陆记录的情况下首次调用该方法将自动登陆，无登陆记录则进入登陆界面，登陆界面可选择注册新的账号等。开发者需要注意进行登陆结果的通知处理。

2.判断玩家登陆状态

函数：

```
boolean tbIsLogined()
```

3.注销

注销账号分两种，一种是有开发者在任何场合调用，以注销当前登陆的账户；另一种是由玩家在用户中心里选择切换当前账号操作。

A) 开发者触发账号注销

注销可选择是否情况用户自动登陆信息，0 表示保留（默认值），1 表示清除.

函数：

```
void tbLogout(Context context, int flag)
```

开发者要注意玩家注销账号后及时进行游戏内注销操作。

B) 玩家触发账号注销 玩家触发是指玩家在个人中心点击切换账号操作，游戏不支持切换账号的，

可以到后台配置不在个人中心显示切换账号（后台配置不切换账号后，点击切换账号不自动打开登陆界面切换账号）。开发者需要通过设置切换账号监听通知来监听账号切换过程，后台配置不切换账号后只有注销成功回调。

函数（本函数为全局函数，多次调用后面设置会覆盖前面设置）：

```
void setOnSwitchAccountListener(OnSwitchAccountListener listener)
```

切换账号回调接口：

```
interface OnSwitchAccountListener {
    /**
     * @param flag   状态码
     * @param user   用户信息，根据状态码改变
     */
     void onSwitchAccountFinished(int flag, TbUser user);
}
```

回调返回结果：

状态码（flag）	描述
SWITCH_ACCOUNT_START	将要切换账号（user 为被注销玩家的信息）
SWITCH_ACCOUNT_SUCCESS	切换账号成功（user 为新玩家的信息）
SWITCH_ACCOUNT_CANCEL	取消切换账号（user 为空）
SWITCH_ACCOUNT_FAILED	切换账号失败（user 为空）

SWITCH_ACCOUNT_LOGOUT	注销账号成功（后台设置不切换账号只有这个回调结果）

4.登录结果通知 玩家进行登录操作并且成功后，平台会发送登录结果通知，开发者需要在这

个通知中进行相应的处理。注意在收到通知后再进行登录成功相关操作，以保证玩家当前的正确登录状态。通知接口的具体代码如下：

```
interface OnLoginFinishedListener {
    void onLoginFinished(boolean success, TbUser user);
}
```

判断登录成功之后可以进行用户信息的获取（强烈建议使用uid 作为游戏角色的唯一标识，account 在特殊环境下可能存在重复的情况，使用 account 作为游戏角色唯一标识所产生的未知后果由游戏方自负）。

```
public class TbUser
    { public int uid;
    public String account;
    public String name;
    public String session;
}
```

*登录成功之后，可以根据 session 向服务器验证当前会话的有效性。 请求为 GET 请求，接口数据传输格式返回是文本格式，返回编码为 UTF8。

检测 Session 是否有效接口说明（从开发者服务器或客户端发起验证都可）：
请求地址：
http://ad.tgi.tongbu.com/api/LoginCheck.ashx?session=afa1c75257f400079a576588d0bb41bc&appid=100000 接口描
述：验证登录信息是否有效 参数说明：
session(SDK 登录成功后获取)
appid(联运后台分配的游戏 ID，每款游戏唯一)
成功：大于 0 的一个数字（UserID）
失败：
0：session 已经过期
-1：格式有错

7.平台离开通知

在关闭所有同步游戏平台 SDK 界面时都会发出该通知 （登录成功后自动离 开平台时不通知）

函数：

```
void tbLeavedSDKPlatform(OnLeavePlatformListener listener)
```

通知接口：

```
interface OnLeavePlatformListener {
    /**
     * @param type    离开类型
     * @param order     订单号（只有从充值页面离开有值）
     */
    void onLeaveFinished(int type, String order);
}
```

返回 type 定义:

对应的状态码	描述
TBPlatformLeavedFromLogin	离开注册、登录页面（登陆成功不发送离开通知）
TBPlatformLeavedFromUserCenter	离开个人中心、系统公告、论坛等
TBPlatformLeavedFromUserPay	离开充值页面（包括成功、失败）
TBPlatformLeavedDefault	离开未知平台(预留状态)

使用方法示例：

```
TbPlatform.getInstance().tbLeavedSDKPlatform(new OnLeavePlatformListener() {
            @Override
            public void onLeaveFinished(int type, String order)
                { switch (type) {
                    case TBPlatformLeavedFromLogin:
                      //从登陆页离开
                        break;
                    case TBPlatformLeavedFromUserCenter:
```

```
                    //从个人中心离开
                    break;
                case TBPlatformLeavedFromUserPay:
                    //从充值页面离开
                    break;
                case TBPlatformLeavedDefault:
                    break;
            }
        }
    });
```

8.虚拟币充值、物品购买

注意：开发者需要在后台配置相关服务器回调通知地址，如图：

SDK 提供两个充值购买相关接口：

1.使用推币进行指定价格的商品购买

函数：

```
/**
 * 进行虚拟币充值或商品购买（需登录，同步后台记录充值账号记录）
 * @param context
 * @param info    购买商品信息
 * @param listener    回调对象
 */
```

```
void tbUniPayForCoin(Context context, TbPayInfo info, OnPayProcessListener listener)
```

TbPayInfo 参数描述：

参数	描述
String serial	合作商订单号，必须保证唯一，双方对账的唯一标记
int needPayRMB	需要支付的金额，单位：元
String payDescription	支付描述，发送支付成功通知时，返回给开发者

说明：该方法为指定价格的物品购买接口，传入的 needPayRMB 为玩家所需要支付的人民币数量（1RMB=10 推币）。购买分为两种情况：一：在推币足够支付该订单时，提示用户直接购买，并通过回调对象 OnPayProcessListener 通知购买结果。

回调结果如下：

```
interface OnPayProcessListener {
    /**
     * 使用推币直接购买成功
     * @param order 订单号
     */
    void onBuyGoodsDidSuccessWithOrder(String order);
    /**
     * 使用推币直接购买商品失败
     * @param order    订单号
     * @param code     错误类型
     */
    void onBuyGoodsDidFailedWithOrder(String order, int code);
    /**
     *   推币余额不足，进入充值页面（开发者需要手动查询订单以获取充值结果）
     * @param order       订单号
     */
    void onBuyGoodsDidStartRechargeWithOrder(String order);
```

```
    /**
     *  跳提示框时，用户取消
     * @param order   订单号
     */
    void onBuyGoodsDidCancelByUser(String order);
}
```

购买失败时错误类型描述：

错误类型	描述
FLAG_PAY_ERROR_QUERY	查询推币余额失败
FLAG_PAY_ERROR_ORDER_EMPTY	订单为空
FLAG_PAY_ERROR_NETWORK	网络异常（有可能已经购买成功，但客户端 已超时，建议进行订单查询）
FLAG_PAY_ERROR_OTHER	其他错误

二：在推币不足以支付该订单时，提示用户进入 Web 充值页进充值购买，此时通过回调对象（OnPayProcessListener）告知客户端已进入充值页面。在退出充值页后需手动调用订单查询接口进行查询。查询接口：

```
    /**
     * 查询订单是否支付成功
     * @param orderId   订单号
     * @param listener  支付结果回调
     */
    void tbCheckPaySuccess(Context mContext, String orderId, OnCheckPayListener listener)
```

查询订单回调接口：

```
interface OnCheckPayListener {
    void CheckOrderFinished(TbOrderInfo tbOrderInfo);
}
```

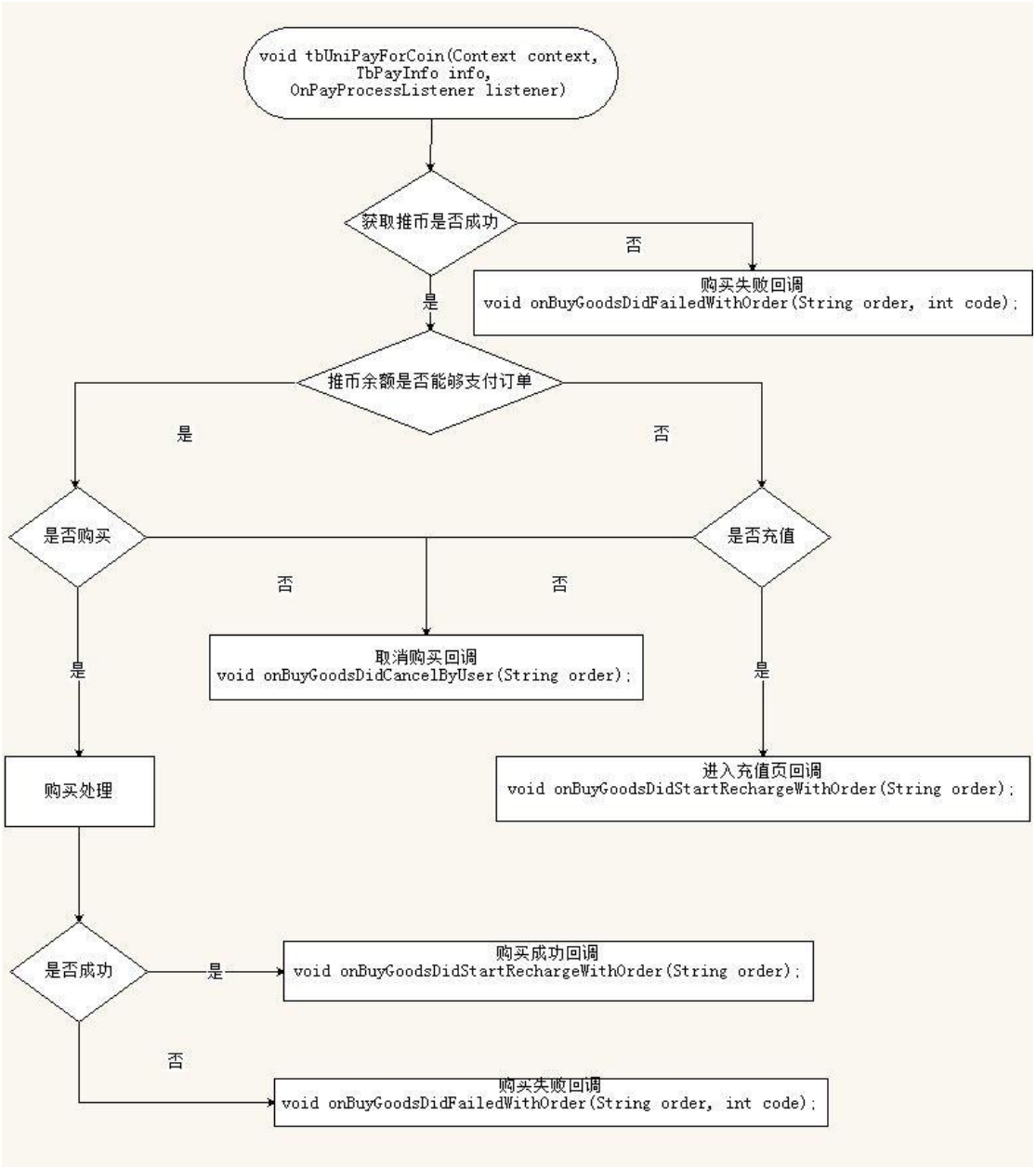
TbOrderInfo 参数描述：

参数	描述
code	状态码
message	返回描述
value	金额
order	订单号

查询结果描述（根据 code 判断）：

状态码	描述
-2	查询失败（订单号为空）
-1	查询失败（网络错误或者服务器返回错误，建议重试）
0	订单未支付，有回调信息
1	订单支付成功，回调成功
2	订单支付成功，回调失败
3	订单未支付，回调失败

支付流程图：



2.玩家自选金额进行虚拟币的充值兑换

函数：

```
/**
 * 进行充值，该方法直接进入 web 页进行充值，无回调，可以通过 tbCheckPaySuccess 进行订单查询
 * @param context
 * @param info   订单描述 （同定额支付，只需设置订单号和支付描述）
 */
void tbUniPayForCoin(Context context, TbPayInfo info)
```

调用该借口后直接进入 Web 充值页，由玩家选择需要充值的数量。建议开发者在退出离开平台的消息处理方法中调用 tbCheckPaySuccess 进行订单查询。

3.测试支付充值

请联系同步运营专员获取测试账号，测试账号中的推币只允许在对应的游戏中进行充值测试，测试账号生成的所有订单不纳入结算。

9.其他平台相关操作

1.进入用户中心 针对使用同步账号的用户，用户可以进行查看当前账号，修改密码，查看充值记录，切换账号，查看帮助等操作。

函数：

void tbEnterUserCenter(Context mContext)	若用户未登录，直接进入登录界面

2.进入游戏论坛

（注：默认论坛不开启，如果需要有论坛，请联系同步运营专员）

函数：

void tbEnterAppBBS(Context mContext)	若用户未登录，直接进入登录界面

3.进入系统公告

函数：

void tbEnterUserMessageBox(Context mContext)	若用户未登录，直接进入登录界面

10. 混淆

tbSDK 包是以 jar 包及资源文件提供给用户的，其中 jar 包已经半混淆状态，您在混淆自己 APK 包的时候请不要将 tbSDK 的 jar 包一起混淆，因为里面有些自定义 UI 控件，若被混淆后会因为无法找到相关类而抛异常。

您可以在用 ant 构建混淆包的 build.xml 里面对应位置或者在 proguard.cfg 里加入：

```
-dontwarn com.alipay.**
-keep class com.alipay.** {*;}

-dontwarn  com.ta.utdid2.**
-keep class com.ta.utdid2.** {*;}

-dontwarn  com.ut.device.**
-keep class com.ut.device.** {*;}

-dontwarn  com.tencent.**
-keep class com.tencent.** {*;}

-dontwarn  com.unionpay.**
-keep class com.unionpay.** {*;}

-dontwarn com.tongbu.sdk.**
-keep class com.tongbu.sdk.** {*;}
```