GAME ANALYTICS

TalkingData 用数据说话

接入指南 3.0.x

最后修正：2014-12-29

目录

一、 综述

1. 适用范围

TalkingData GameAnalytics 帮助游戏开发者解决玩家数据收集至数据标准化分析的全部繁琐问题，以行业标准指标形式将数据展现于报表中。

本指南适用于使用 cocos2D-x 开发游戏，同时支持 iOS 和 Android 平台。
注意:由于 Cocos2dx-3.0 以上的使用方式跟之前不同，建议参照 TalkingData 提供的 cocos3.0 的 demo 集成 。

2. 统计标准

为了与游戏自有体系更好的结合，统计中我们采用游戏自身的帐户来做为一个玩家单元。数据中除了玩家基础信息外，主要帮助处理游戏过程中的升级、任务、付费、消费等详细行为数据。行为标准可参考以下说明：

- 玩家

  游戏自身的帐户，通常为一个唯一号、唯一名或一份唯一存档号。是平台中计算数据的最基础单元。

- 设备

  指一台安装了游戏包的终端。

- 玩家的一次游戏

  玩家从打开游戏界面至离开游戏界面的完整过程，如果玩家在离开游戏界面后 30 秒内重新回到游戏中，将被认为是上次游戏被打扰后的延续，记为一次完整游戏。

- 付费

  特指玩家充值现金换取虚拟币的过程。充值的现金将作为收入进行统计。

- 消费

  指玩家在一个消费点上消耗虚拟币的过程。

二、 接入流程

STEP 1、为游戏申请 APP ID

进入 talkinggame.com 网站，使用您的注册账号登录后，请预先创建一款游戏，您将获得一串 32 位的 16 进制 APP ID，用于唯一标识您的一款游戏。

STEP 2、解压缩

解压缩 SDK 包当中的 Game_Analytics_SDK_Cocos.zip 文件，解压目录建议放置在 cocos2d-x 的平级目录下，对于配置一些路径会方便很多。

STEP 3、导入依赖工程

此步骤根据开发者文件目录的不同，操作会有不同，以下是在默认路径下的操作流程。如果您在该步骤遇到问题，请及时联络技术支持寻求帮助。

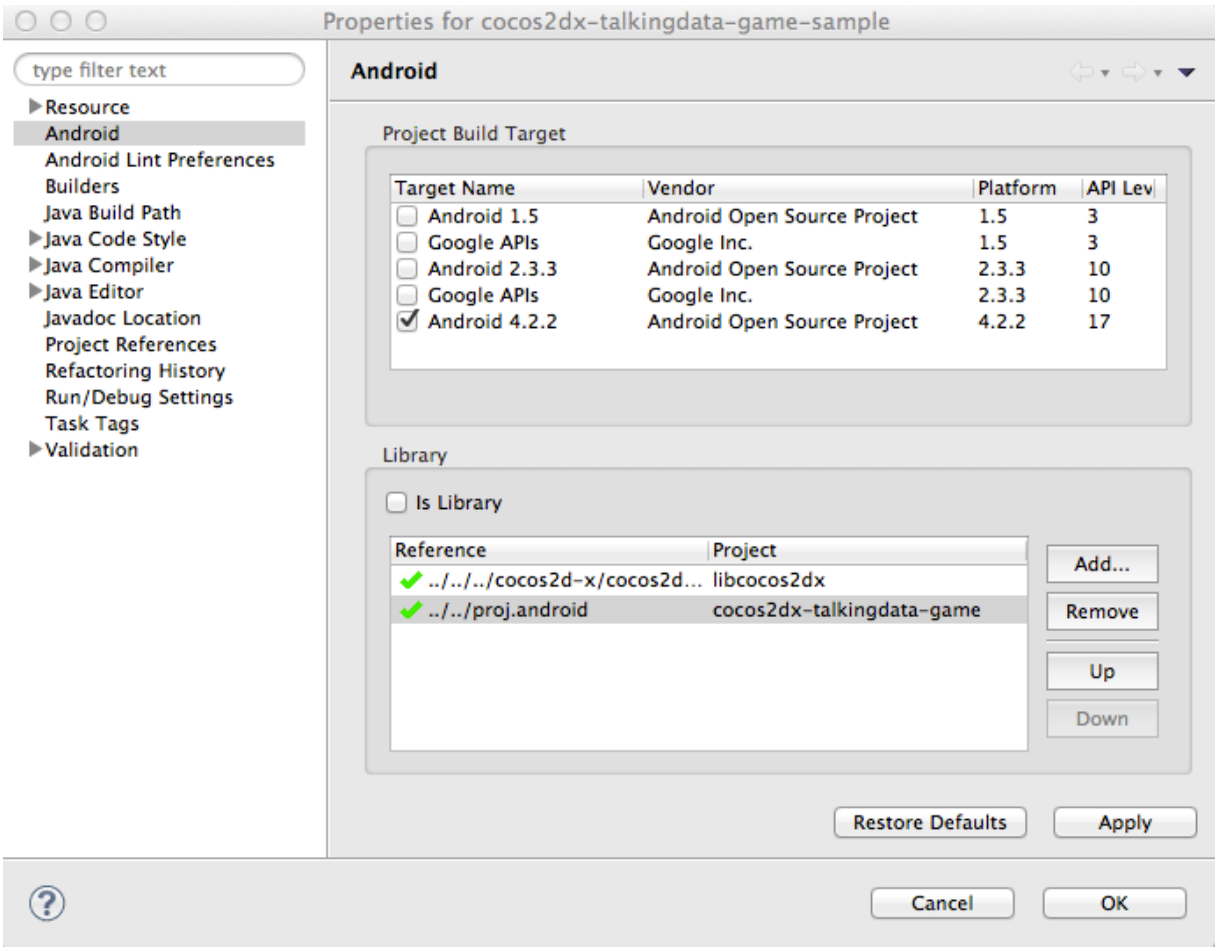
联络邮箱：support@tendcloud.com　QQ 支持：4008701230

A). Android 平台

1. 在 Eclipse 里选择 File->Import 选择 cocos2dx-talkingdata/proj.android 目录，并导入到 Eclipse 中。

2. 在游戏工程中添加对上一步引入的工程的依赖。

选中工程点 File->Preperties->android



3. 修改游戏项目中的 Android.mk。

LOCAL_C_INCLUDES 添加依赖工程中的 include 目录.

LOCAL_WHOLE_STATIC_LIBRARIES 添加依赖工程编译的 module 名字.

可以参考 cocosdx-talkingdata/sample/proj.android

4. 配置 JNI 的 java 环境 – 必须。

需要在 JNI_OnLoad 方法中初始化 jni 的 java 环境 ,请在该方法中加入以下调用：

```
TDGAJniHelper::setJavaVM(vm);
```

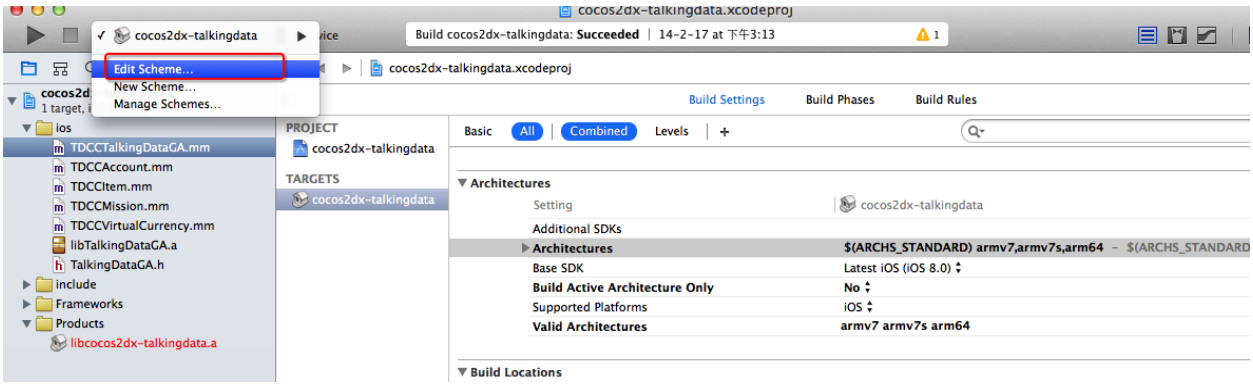
此步骤是 C 语言可调用 java 的关键，未进行调用会导致所有数据无法记录，请 Android 开发者注意。

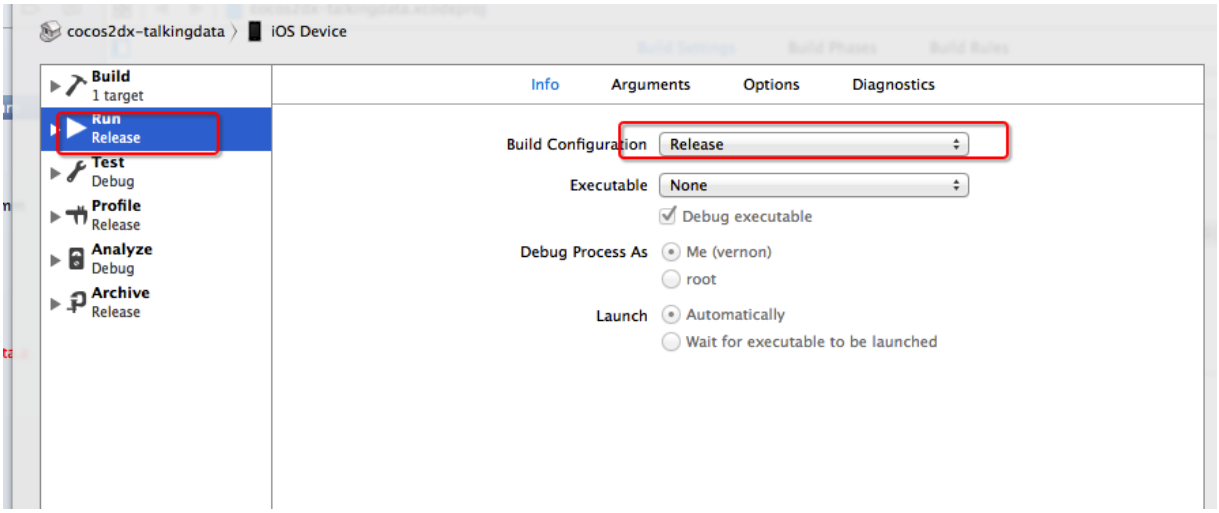
如果您用的是 Cocos2dx-3.0，可能在 main.cpp 里不再用 JNI_OnLoad 方法，那么只需要在 cocos_android_app_init 方法中的最开始的地方添加。

b). iOS

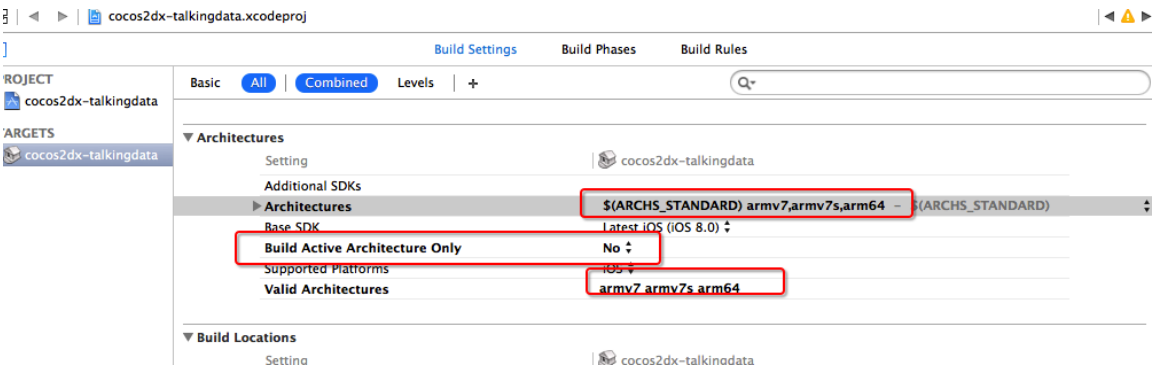
1、直接打开 game-analytic-cocos2dx 里的 cocos2dx-talkingdata.xcodeproj 这个工程.

2、修改 edit scheme，为 release 模式。如图：



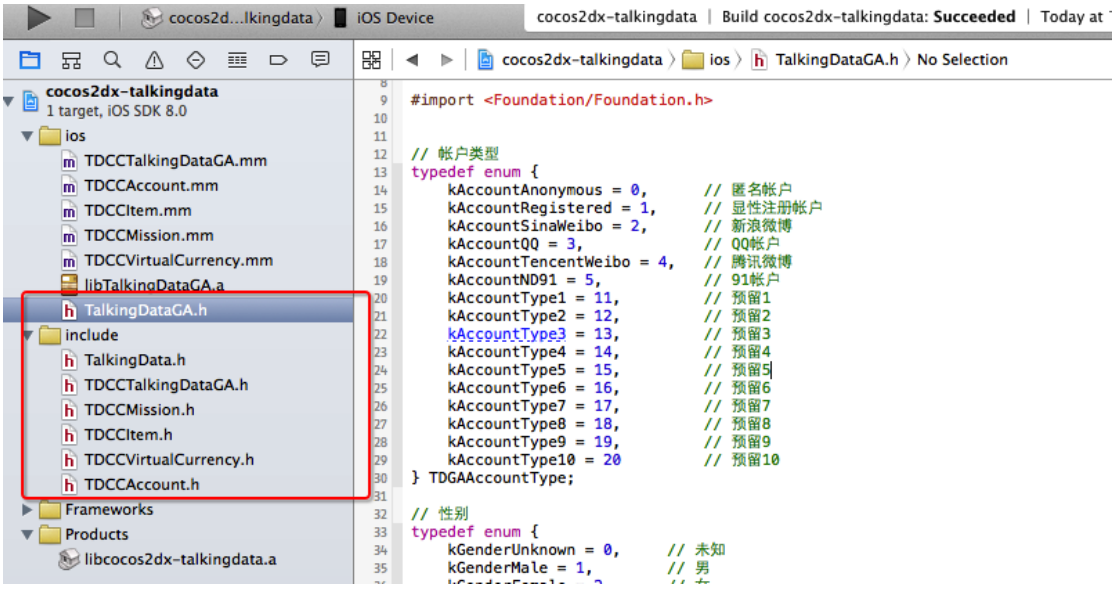


3、修改编译参数，把自己想要支持的架构都添加上。如图：



4、Command+B 开始编译（这里会生成模拟器和真机两个版本的.a 库）。

5、然后把 Products 文件夹中生成的 libcocos2dx-talkingdata.a 的库和 SDK 文件中的 include 文件夹中的.h 文件和 TalkingDataGA.h 都拖拽到您的游戏工程中。



STEP 4、配置 ANDROIDMANIFEST.XML(仅限 ANDROID 平台)

- 在ANDROID平台上，SDK需要获取适当的权限才可以正常工作；开发者需要在AndroidManifest.xml里边添加下表列举出来的所有权限申明。

游戏需要的权限	用途
INTERNET	允许程序联网和发送统计数据的权限。
ACCESS_NETWORK_STATE	允许游戏检测网络连接状态，在网络异常状态下避免数据发送，节省流量和电量。
READ_PHONE_STATE	允许游戏以只读的方式访问手机设备的信息，通过获取的信息来定位唯一的玩家。
ACCESS_WIFI_STATE	用来获取设备的 mac 地址。
WRITE_EXTERNAL_STORAGE	用于保存设备信息，以及记录日志。
ACCESS_FINE_LOCATION（可选）	用来获取该游戏被使用的精确位置信息。
ACCESS_ COARSE_LOCATION（可选）	用来获取该游戏被使用的粗略位置信息。
GET_TASKS(必须)	注：这个是新添加进来的权限

示例代码：

```
<?xml version="1.0" encoding="utf-8"?>
```

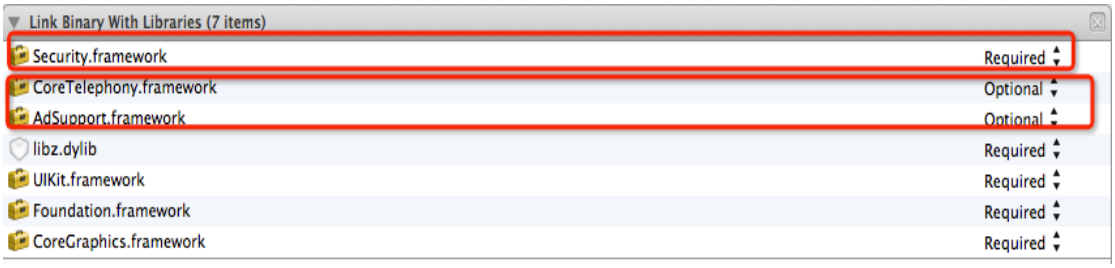
```
<manifest ……>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission
android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"
/>
    <uses-permission android:name="android.permission.GET_TASKS" />
    <application ……>

    </application>
</manifest>
```

STEP 5、添加依赖的框架(仅限 iOS 平台)

- TalkingData Game Analytics 需要使用 Security. framework 来辅助存储设备标识，CoreTelephony.framework 框架获取运营商标识，使用 AdSupport.framework 获取 advertisingIdentifier，使用 libz.dylib 进行数据压缩。 Xcode 的添加方式如下所示：

xcode 版本	操作
Xcode6.x	在您的工程里，选择 target-->Build Phases-->Link Binary With Libraries，点击 + 号，选择 Security.framework 、 CoreTelephony.framework 、 AdSupport.framework(仅 xcode4.5 及以上版本添加)和 libz.dylib

依赖条件Security.framework必须为Required，其他可以是Optional，如下图：



STEP 6、添加调用方法

参考“三、添加调用方法“的指导来完成开发。

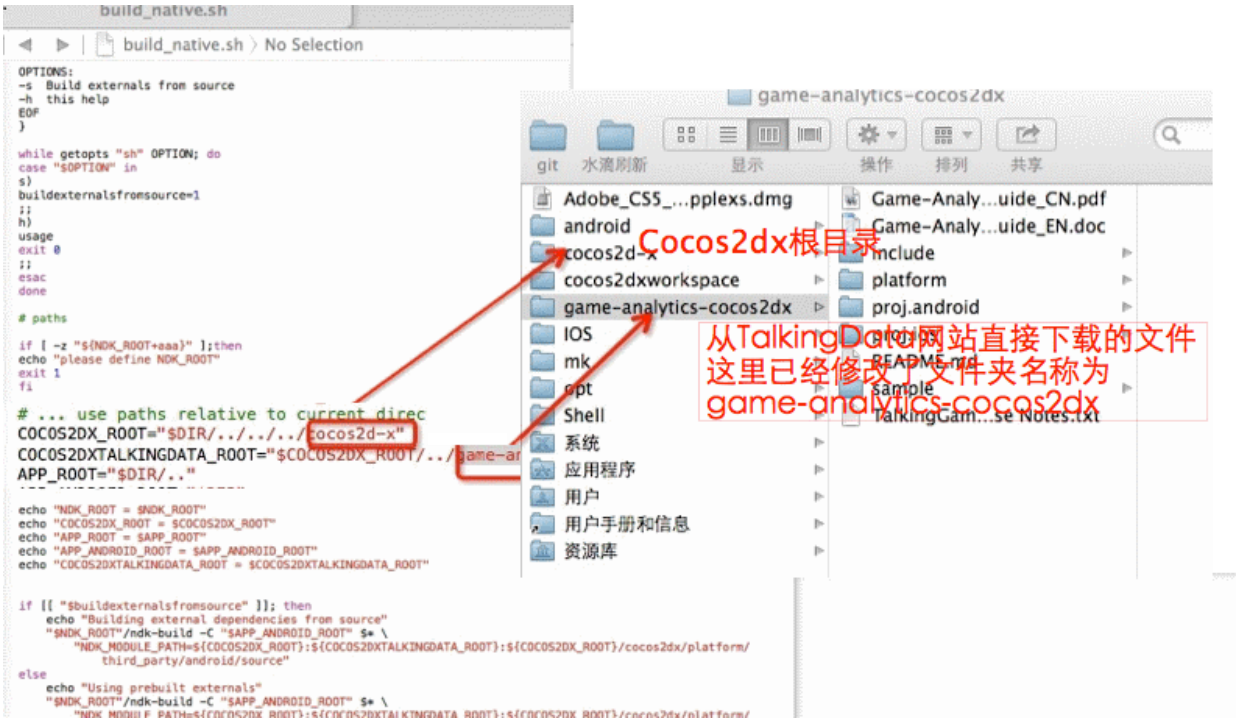
STEP 7、进行数据测试

调用方法添加完毕后，应当对游戏打包并进行数据测试，以确保打包的正确。

注意

1、（coco2dx3.0 以前版本）运行 Demo 应用，一般只需要修改两个地方，

如图：



2、(cocos2dx3.0 版本)不再使用了 build_native.sh，转用了

build_native.py,进行修改如下：

```
def do_build(cocos_root, ndk_root, app_android_root,ndk_build_param,sdk_root,android_platform,build_mode):

    ndk_path = os.path.join(ndk_root, "ndk-build")

    # windows should use ";" to seperate module paths
    platform = sys.platform
    if platform == 'win32':
        ndk_module_path = 'NDK_MODULE_PATH=%s;%s/external;%s/cocos' % (cocos_root, cocos_root, cocos_root)
    else:
        ndk_module_path = 'NDK_MODULE_PATH=%s:%s/external:%s/cocos:/Users/vernon/game-analytics-cocos2dx' % (cocos_root, cocos_root, cocos_root)

    num_of_cpu = get_num_of_cpu()

    if ndk_build_param == None:
        command = '%s -j%d -C %s %s' % (ndk_path, num_of_cpu, app_android_root, ndk_module_path)
    else:
        command = '%s -j%d -C %s %s %s' % (ndk_path, num_of_cpu, app_android_root, ''.join(str(e) for e in ndk_build_param), ndk_module_path)
    if os.system(command) != 0:
        raise Exception("Build dynamic library for project [ " + app_android_root + " ] fails!")
```

只需要在这个里面添加指向 TalkingData SDK 的路径可以。如果是在 windows 下编译请在上面的位置添加。具体请参考 TDCocosGame3.0。或者找 TalkingData 客服人员进行询问。

3、(cocos2dx3.2 之后版本)到了 3.2 之后版本，只需要修改 Android.mk 文件就可以了。如图：

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

$(call import-add-path,$(LOCAL_PATH)/../../cocos2d)
$(call import-add-path,$(LOCAL_PATH)/../../cocos2d/external)
$(call import-add-path,$(LOCAL_PATH)/../../cocos2d/cocos)
$(call import-add-path,$(LOCAL_PATH)/../../../game-analytics-cocos2dx)

LOCAL_MODULE := cocos2dcpp_shared

LOCAL_MODULE_FILENAME := libcocos2dcpp

LOCAL_SRC_FILES := hellocpp/main.cpp \
                   ../../Classes/AppDelegate.cpp \
                   ../../Classes/AccountScene.cpp

LOCAL_C_INCLUDES := $(LOCAL_PATH)/../../Classes \
$(LOCAL_PATH)/../../../game-analytics-cocos2dx/include \
$(LOCAL_PATH)/../../../game-analytics-cocos2dx/platform/android

LOCAL_WHOLE_STATIC_LIBRARIES := cocos2dx_static
LOCAL_WHOLE_STATIC_LIBRARIES += cocos2dx-talkingdata
LOCAL_WHOLE_STATIC_LIBRARIES += cocosdenshion_static

# LOCAL_WHOLE_STATIC_LIBRARIES += box2d_static
# LOCAL_WHOLE_STATIC_LIBRARIES += cocosbuilder_static
# LOCAL_WHOLE_STATIC_LIBRARIES += spine_static
# LOCAL_WHOLE_STATIC_LIBRARIES += cocostudio_static
# LOCAL_WHOLE_STATIC_LIBRARIES += cocos_network_static
# LOCAL_WHOLE_STATIC_LIBRARIES += cocos_extension_static


include $(BUILD_SHARED_LIBRARY)

$(call import-module,.)
$(call import-module,audio/android)
$(call import-module,proj.android/jni)

# $(call import-module,Box2D)
# $(call import-module,editor-support/cocosbuilder)
# $(call import-module,editor-support/spine)
# $(call import-module,editor-support/cocostudio)
```

指名cocos2dx-talkingdata这个module的路径
原来在build_native.py里进行设置

添加自己所使用的cpp文件

添加使用talkingdata所必须的头文件

添加对module的引用

调用编译方法

三、 添加调用方法

1. 游戏启动和关闭

用途和用法

- 用于准确跟踪玩家的游戏次数，游戏时长等信息。
- Android

- 在启动的 Activity 的 onCreate()方法中添加对 TalkingDataGA.init(Context ctx, String appId, String channelId)的调用。Init 方法会初始化 SDK，在此之前，不可以调用 SDK 的其他方法。
- 如果使用 TalkingData 的营销策略的话必须在：AndroidManifest.xml 里边有配置自定义的 Application 类，也可以在 Application 的 onCreate()方法里调用 init 来初始化 SDK。具体代码修改参考 四、营销策略。
- 在启动的 Activity 中的 onResume()方法中添加对 TalkingDataGA.onResume(Activity act)的调用，在 onPause()方法里添加对 TalkingDataGA.onPause(Activity act)的调用。

➢ iOS

- 游戏启动时调用 TDCCTalkingDataGA::onStart(const char* appID, const char* channelId)
- 可以在 channelId 中填入推广渠道的名称，数据报表中则可单独查询到他们的数据。每台设备仅记录首次安装激活的渠道，更替渠道包安装不会重复计量。不同的渠道 ID 包请重新编译打包。
- 用营销策略的注意啦：之前在 IOS 中 初始化都会放在 AppDelegate.cpp 的 applicationDidFinishLaunching 方法中，如果您想使用营销策略，需要把初始化的代码放在 Native 中 AppController.mm 的代码中：

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    // Override point for customization after application launch.

    // Add the view controller's view to the window and display.
    window = [[UIWindow alloc] initWithFrame: [[UIScreen mainScreen] bounds]];
    EAGLView *__glView = [EAGLView viewWithFrame: [window bounds]
                                     pixelFormat: kEAGLColorFormatRGBA8
                                     depthFormat: GL_DEPTH_COMPONENT16
                              preserveBackbuffer: NO
                                                                  sharegroup:nil
                                                               multiSampling:NO
                                                             numberOfSamples:0];

    // Use RootViewController manage EAGLView
    viewController = [[RootViewController alloc] initWithNibName:nil bundle:nil];
    viewController.wantsFullScreenLayout = YES;
    viewController.view = __glView;

    // Set RootViewController to window
    if ( [[UIDevice currentDevice].systemVersion floatValue] < 6.0)
    {
        // warning: addSubView doesn't work on iOS6
        [window addSubview: viewController.view];
    }
    else
    {
        // use this method on ios6
        [window setRootViewController:viewController];
    }

    [window makeKeyAndVisible];

    [[UIApplication sharedApplication] setStatusBarHidden: YES];
    [[UIApplication sharedApplication]registerForRemoteNotificationTypes:UIRemoteNotificationTypeAlert|UIRemoteNotificationTypeBadge|
        UIRemoteNotificationTypeSound];

    [TalkingDataGA onStart:@"2B002F9CD724EB09730AF32CB4D909C3" withChannelId:@"Appstore"];
    if (![TalkingDataGA handleTDGAPushMessage:launchOptions]) {
        // other code
    }
```

具体代码修改参考 四、营销策略。

接口说明：（TDCCTalkingDataGA 类）

```
static void onStart(const char* appID, const char* channelId);  (只用于 iOS)
```

参数说明：

参数	类型	描述
appId	char	填写创建游戏时获得的 App ID，用于唯一识别您的游戏。
channelId	char	用来标注游戏推广渠道，区分玩家的来源来查看统计。 格式：32 个字符内，支持中文、英文、数字、空格、英文点和下划线，请勿使用其他符号。

示例代码：

```
TDCCTalkingDataGA::onStart("ABCDEF0123456", "APPLE_APP_STORE");
```

2. 统计玩家帐户

用途和用法

- 定义一个玩家单元，更新玩家最新的属性信息。
- 在可确定玩家帐户后尽早调用 TDCCAccount::setAccount(const char* accountId)方法来传入 accountId ,对一个玩家的分析依赖于这个 accountId。
- accountId 需保证全局唯一，且终生不变。在多设备中传入同样 accountId，数据将归入同一个玩家内，玩家数不增加；同一设备中传入多个不同 accountID，计算多个玩家账户数。
- 玩家的等级、年龄性别等属性发生变化时，尽快调用对应 set 方法来更新帐户属性。每次玩家登录或变更区服后，游戏通常会同步玩家游戏资料，在同步完成时也需调用 setLevel，以确保 Level 准确，数据更精准。
- 如果期望以设备为玩家单元进行统计，请在 setAccount 时传入 TDCCTalkingDataGA::getDeviceId()，SDK 将帮助自动定义设备 ID 作为 Account，全部数据指标也会以设备 ID 为依据进行计算，详细参考示例 3。

接口说明：（TDCCAccount 类**）**

```
//返回用户对象
static TDCCAccount* setAccount(const char* accountId)
//设置帐户类型
```

```
void setAccountType (TDCCAccountType accountType)
//设置帐户的显性名
void setAccountName(const char* accountName)
//设置级别
void setLevel (int level)
//设置性别
void setGender (TDCCGender gender)
//设置年龄
void setAge (int age)
//设置区服
void setGameServer（const char* gameServer)
```

参数说明：

参数	类型	描述
accountId	char	设定帐户唯一标识，用于区分一个玩家，最多 64 个字符。其他调用依赖于此 ID。 *如果无玩家账户或期望以设备为单位计算玩家，调用时传入 TDCCTalkingDataGA::getDeviceId() 即可。
accountType	TDCCAccountType	传入帐户的类型。系统预定义的类型为： 匿名: TDCCAccountType. kAccountAnonymous 自有帐户显性注册：TDCCAccountType. kAccountRegistered 国内主流的第三方帐号: 新浪微博：TDCCAccountType. kAccountSinaWeibo QQ：TDCCAccountType. kAccountQQ 腾讯微博：TDCCAccountType. kAccountTencentWeibo 网龙 91：TDCCAccountType. kAccountND91 另外系统预留了 10 种自定义的帐户类型，分别为 TDCCAccountType. kAccountType1 到 TDCCAccountType. kAccountType10
accountName	char	在帐户有显性名时，可用于设定帐户名，最多支

		持 64 个字符。
Level	int	设定玩家当前的级别，未设定过等级的玩家默认的初始等级为“1”，支持的最大级别为 1000。 *等级发生变化时尽快进行调用。 *每次玩家登录、换区服，游戏通常会同步玩家资料，同步完成时需调用 setLevel，可使等级数据精准。 （伴随 setAccount 调用，如等级可确定，都建议在其后调用一下 setLevel。）
gender	TDCCGender	设定玩家性别： 男: TDCCGender. kGenderMale 女: TDCCGender. kGenderFemale 未知: TDCCGender. kGenderUnknown
age	int	设定玩家年龄，范围为 0-120。
gameServer	char	传入玩家登入的区服，最多 16 个字符。

示例 1：

如一个游戏内 UID 为 10000 的匿名游戏玩家以匿名（快速登录）方式在国服 2 区进行游戏，并在游戏中由 1 级升至 2 级，之后玩家使用 QQ 号 5830000 在游戏中进行显性注册，并设定为 18 岁男玩家的整个过程。

```
TDCCAccount* account = TDCCAccount::setAccount("10000");
account->setAccountType(kAccountAnonymous);
account->setLevel(1);
account->setGameServer("国服 2");
```

在玩家升级时，做如下调用

```
account->setLevel(2);
```

在玩家显性注册成功时做如下调用

```
account->setAccountName("5830000@qq.com");
account->setAccountType(kAccountAnonymous);
account->setAge(18);
account->setGender(kGenderMale);
```

示例 2：

如一款必须在注册后进行的不分区服的游戏中，玩家使用其已注册的帐号：xiaoming@163.com（游戏服务器中为玩家分配了 101111 的内部号）登录进行游戏。

```
TDCCAccount* account = TDCCAccount::setAccount("101111");
account->setAccountName("xiaoming@163.com");
account->setAccountType(kAccountAnonymous);
```

示例 3：

如在一款类似愤怒小鸟的休闲游戏中，玩家直接进入进行游戏。

您可自行决定唯一 ID 规则，如设备 IMEI、MAC 地址等，也可以使用 TalkingData 提供的设备唯一 ID 接口，例如：

```
TDCCAccount* account =
TDCCAccount::setAccount(TDCCTalkingDataGA::getDeviceId());
account->setAccountType(kAccountAnonymous);
```

3. 跟踪玩家充值

用途和用法

- 跟踪玩家充值现金而获得虚拟币的行为，充入的现金将反映至游戏收入中。
- 充值过程分两个跟踪阶段：1、发出有效的充值请求；2 确认某次充值请求已完成充值。

  您可在玩家发起充值请求时（例如玩家选择了某个充值包，进入支付流程那一刻）调用 onChargeRequest，并传入该笔交易的唯一订单 ID 和详细信息；在确认玩家支付成功时调用 onChargeSuccess ,并告知完成的是哪个订单 ID。

注意：

1、orderID 是标识交易的关键，每一次的充值请求都需要是不同的 orderID，否则会被认为重复数据而丢弃，造成收入数据偏差的情况。

2、orderID 是您自己构造的订单 ID，可以使用类似 userID+时间戳+随机数 的方式来自己定义 orderID，来保障其唯一性。

3、收入数据以调用了 onChargeSuccess 为准，平台按 Success 调用时传入的 orderID 来追溯对应的 onChargeRequest 中的金额来确定充值成功的金额，Request 必须调用，且需要早于 Success，否则可能影响收入数据的金额计数。

接口说明：（TDCCVirtualCurrency 类）

```
//充值请求
static void onChargeRequest(const char* orderId, const char* iapId, double currencyAmount, const char* currencyType, double virtualCurrencyAmount, const char* paymentType)
//充值成功
static void OnChargeSuccess (const char* orderId)
```

参数说明：

参数	类型	描述
orderId	char	订单 ID，自行构造，最多 64 个字符。 用于唯一标识一次交易。 *如果多次充值成功的 orderID 重复，将只计算首次成功的数据，其他数据会认为重复数据丢弃。 *如果 Success 调用时传入的 orderID 在之前 Request 没有对应 orderID，则只记录充值次数，但不会有收入金额体现。
iapId	char	充值包 ID，最多 32 个字符。 例如：VIP3 礼包、500 元 10000 宝石包
currencyAmount	double	现金金额或现金等价物的额度
currencyType	char	请使用国际标准组织 ISO 4217 中规范的 3 位字母代码标记货币类型。点击查看参考 例：人民币 CNY；美元 USD；欧元 EUR （如果您使用其他自定义等价物作为现金，亦可使用 ISO 4217 中没有的 3 位字母组合传入货币类型，我们会在报表页面中提供汇率设定功能）
virtualCurrencyAmount	double	虚拟币金额
paymentType	char	支付的途径，最多 16 个字符。 例如：“支付宝” “苹果官方” “XX 支付 SDK”

示例 1：

玩家使用支付宝方式成功购买了“大号宝箱”（实际为 100 元人民币购入 1000 元宝的礼包），该笔操作的订单编号为 order001。可以如下调用：

1）在向支付宝支付 SDK 发出请求时，同时调用：

```
TDCCVirtualCurrency::onChargeRequest("order001", "大号宝箱", 100, "CNY", 1000, "AliPay");
```

2）订单 order001 充值成功后调用：

```
TDCCVirtualCurrency::onChargeSuccess("order001");
```

示例 2：

在一款与 91 联运的游戏中，游戏使用了 91 的支付聚合 SDK，玩家购买一个“钻石礼包 1”(10 个 91 豆购买 60 钻石),该笔操作的订单号为“7837331”。由于此类聚合 SDK 往往要求使用其自有的“代币”（91 使用 91 豆，兑换人民币比例 1：1）做充值依据，建议将“代币”折算为人民币后再调用统计：

```
1）在向 91 支付 SDK 发出请求时，进行调用
TDCCVirtualCurrency::onChargeRequest(“7837331”，"钻石礼包 1", 10, "CNY", 60, "91 SDK");
2）订单 order001 充值成功：
TDCCVirtualCurrency::onChargeSuccess("7837331");
```

4. 跟踪获赠的虚拟币（可选）

用途和用法

- 游戏中除了可通过充值来获得虚拟币外，可能会在任务奖励、登录奖励、成就奖励等环节免费发放给玩家虚拟币，来培养他们使用虚拟币的习惯。开发者可通过此方法跟踪全部免费赠予虚拟币的数据。
- 在成功向玩家赠予虚拟币时调用 onReward 方法来传入相关数据。
- 只获得过赠予虚拟币的玩家不会被记为付费玩家。赠予的虚拟币会计入到所有的虚拟币产出中，也计入到留存虚拟币中。

接口说明：（TDGAVirtualCurrency 类）

```
//赠予虚拟币
public static void onReward (double virtualCurrencyAmount, String reason)
```

参数说明：

参数	类型	描述
virtualCurrencyAmount	double	虚拟币金额。
reason	String	赠送虚拟币原因/类型。 格式：32 个字符内的中文、空格、英文、数字。 不要带有任何开发中的转义字符，如斜杠。 注意：最多支持 100 种不同原因。

示例 1：

玩家在完成了新手引导后，成功获得了免费赠送的 5 个钻石：

```
TDGAVirtualCurrency::onReward(5, "新手奖励");
```

示例 2：

玩家在游戏竞技场中排名较高，而获得了 100 消费券奖励：

```
TDGAVirtualCurrency::onReward(100, "竞技场 Top2");
```

5. 跟踪游戏消费点

用途和用法

- 跟踪游戏中全部使用到虚拟币的消费点，如购买虚拟道具、VIP 服务、复活等
- 跟踪某物品或服务的耗尽
- 在任意消费点发生时尽快调用 onPurchase，在某个道具/服务被用掉（消失）时尽快调用 onUse
- 消费点特指有价值的虚拟币的消费过程，如果游戏中存在普通游戏金币可购买的虚拟物品，不建议在此处统计。

接口说明：（TDCCItem 类）

```
//记录付费点
static void onPurchase(const char* item, int itemNumber, double
priceInVirtualCurrency)
//消耗物品或服务等
static void onUse(const char* item, int itemNumber)
```

参数说明：

参数	类型	描述
item	char	某个消费点的编号，最多 32 个字符。
itemNumber	int	消费数量
priceInVirtualCurrency	double	虚拟币单价

示例 1：

玩家以 25 元宝/个的单价购买了两个类别号为“helmet1”的头盔，可以调用：

```
TDCCItem::onPurchase("helmet1", 2, 25);
```

其中一个头盔在战斗中由于损坏过度而消失。

```
TDCCItem::onUse("helmet1", 1);
```

示例 2：

玩家在某关卡中死亡，使用 5 个钻石进行复活。可调用：

```
TDCCItem::onPurchase("revival", 1, 5);
```

6. 任务、关卡或副本

用途和用法

- 跟踪玩家任务/关卡/副本的情况。
- 同一个 missionId 如果在未结束前，重复进行了 onBegin 调用，则重新开始计时，上一次的调用被丢弃。
- 如果多个不同的 MissionID 同时在进行（都调用了开始，但并未完成或失败），他们都会同时进行计时，而不是只有一个计时其他暂停计时。

接口说明：（TDCCMission 类）

```
//接受或进入
static void onBegin(const char* missionId)
//完成
static void onCompleted(const char* missionId)
//失败
static void onFailed(const char* missionId, const char* cause)
```

参数说明：

参数	类型	是否必填	描述
missionId	char	必填	任务、关卡或副本的编号，最多 32 个字符。此处可填写 ID，别名可在报表编辑。
cause	char	必填	失败原因 ,最多 16 个字符。共支持 100 种原因。

示例 1：

玩家进入名称为“蓝色龙之领地”的关卡。可调用：

```
TDCCMission::onBegin("蓝色龙之领地");
```

玩家成功打过了关卡：

```
TDCCMission::onCompleted("蓝色龙之领地");
```

示例 2：

玩家接到了“主线任务 5”后，又接受了某个直线任务“赚钱 1”，之后他在赚钱任务 1 进行中因为觉得任务过难，放弃任务而失败。

```
TDCCMission::onBegin("主线任务 5");
TDCCMission::onBegin("赚钱 1");
TDCCMission::onFailed("赚钱 1"， "quit");
```

7. 自定义事件

用途和用法

- 用于统计任何您期望去跟踪的数据，如：点击某功能按钮、填写某个输入框、触发了某个广告等。
- 可以自行定义 eventId，在游戏中需要跟踪的位置进行调用，注意 eventId 中仅限使用中英文字符、数字和下划线，不要加空格或其他的转义字符。
- 除了可以统计某自定义 eventId 的触发次数外，还可以通过 key-value 参数来对当时触发事件时的属性进行描述。如定义 eventId 为玩家死亡事件，可添加死亡时关卡、死亡时等级、死亡时携带金币等属性，通过 key-value 进行发送。
- 每款游戏可定义最多 200 个不同 eventId，每个 eventId 下，可以支持 20 种不同 key 的 500 种不同 value 取值（char 类型），并且注意每个单次事件调用时，最多只能附带 10 种不同 key。
- EventParamMap 的 Value 目前仅支持字符串（String）和数字（Number）类型，目前 Number 类型只能以 String 的形式传入如”10”，并且一次事件最多允许附带10种参数。如果value为字符串，TalkingData会统计每种value出现的次数；如果为数字类型，那么 TalkingData 会统计 value 的总和/平均值。对于传入的其他类型，会通过调用转换为字符串进行统计。
- eventId、EventParamMap 的 key 和 String 类型的 value，分别最多支持 32 个字符。

接口说明

- 在游戏程序的 event 事件中加入下面格式的代码，也就成功的添加了一个简单的事件到您的游戏程序中了：

```
static void onEvent(const char* eventId, EventParamMap* map =
NULL);
```

示例 1：

使用自定义事件跟踪玩家的死亡情况，并记录死亡时的等级、场景、关卡、原因等信息，可在玩家死亡时调用：

```
EventParamMap paramMap;
paramMap.insert(EventParamPair("level", "50-60"));
paramMap.insert(EventParamPair("map", "沼泽地阿卡村"));
paramMap.insert(EventParamPair("mission", "屠龙副本"));
paramMap.insert(EventParamPair("coin", "PK 致死"));
TDCCTalkingDataGA::onEvent("dead", &paramMap);
```

注：在某 key 的 value 取值较离散情况下，不要直接填充具体数值，而应划分区间后传入，否则 value 不同取值很可能超过平台最大数目限制，而影响最终展示数据的效果。

如：示例中金币数可能很离散，请先划分合适的区间。

示例 2：

使用自定义事件跟踪玩家在注册过程中每个步骤的失败情况：

```
void XXXX::onClick() {
EventParamMap paramMap;
paramMap.insert(EventParamPair("step", "1"));
TDCCTalkingDataGA::onEvent("reg", &paramMap); // 在注册环节的每一步完成时，以步骤名作为 value 传送数据
}
```

四、 **营销策略**（如果不使用营销策略可以直接跳过）

1. ANDROID：

1) 正确集成第三方推送平台

TalkingData 目前支持四家推送平台，包括：百度、个推、魔推、极光，这四家可单独使用，也可以一起使用。

注意：集成之后需要先测试是否可以接收到第三方推送过来的消息，再进行下一步操作。

2) 添加TalkingDataGameAnalytics运营所必须的Activity。

```
    <activity android:name="com.tendcloud.tenddata.AlertActivity"
android:theme="@android:style/Theme.Translucent.NoTitleBar"  />
```

3) 添加TalkingDataGameAnalytics所必须的BroadCastReceiver。

```
<receiver
android:name="com.tendcloud.tenddata.TalkingDataMessageReceiver"
android:enabled="true" >
    <intent-filter>
        <!-- 必须添加 -->
        <action android:name="com.talkingdata.notification.click"
/>
        <action android:name="com.talkingdata.message.click" />
    </intent-filter>

    <intent-filter>
        <!-- 如果使用极光推送，必须添加 -->
        <action
android:name="cn.jpush.android.intent.REGISTRATION" />
        <action
android:name="cn.jpush.android.intent.MESSAGE_RECEIVED" />
        <category android:name="com.talkingdata.push" />
    </intent-filter>
    <intent-filter>
        <!-- 如果使用百度推送必须添加 -->
        <action
android:name="com.baidu.android.pushservice.action.MESSAGE" />
        <action
android:name="com.baidu.android.pushservice.action.RECEIVE" />
    </intent-filter>
    <intent-filter>
        <!-- 如果使用个推必须添加 ，注：H0DPYSxUkR9NFoWnvff656要换成开发
```

```
者自己的appid-->
        <action
android:name="com.igexin.sdk.action.H0DPYSxUkR9NFoWnvff656" />
    </intent-filter>
    <intent-filter>
        <!-- 如果使用MPUsh推送必须添加 -->
        <action
android:name="android.mpushservice.action.media.MESSAGE" />
        <action
android:name="android.mpushservice.action.media.TOKEN" />
    </intent-filter>
</receiver>
```

2. iOS：

注册 Notification 类型：

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    [TalkingDataGA onStart:@"APP_ID" withChannelId:nil];
    if ([application respondsToSelector:
        @selector(isRegisteredForRemoteNotifications)]){
      // iOS 8 Notifications
      [application registerUserNotificationSettings:
        [UIUserNotificationSettings
          settingsForTypes:(UIUserNotificationTypeSound |
                            UIUserNotificationTypeAlert |
                            UIUserNotificationTypeBadge)
          categories:nil]];
      [application registerForRemoteNotifications];
  }else{
      // iOS < 8 Notifications
    [application registerForRemoteNotificationTypes:
              (UIRemoteNotificationTypeBadge |
               UIRemoteNotificationTypeAlert |
```

```
                                UIRemoteNotificationTypeSound)];
    }
        // other code
}
```

在 application: didRegisterForRemoteNotificationsWithDeviceToken:方法中调用 setDeviceToken 传入 DeviceToken。

```
- (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken
{
    [TalkingDataGA setDeviceToken:deviceToken];
}
```

分别在 application: didRegisterForRemoteNotificationsWithDeviceToken:和 application:didReceiveRemoteNotification:方法中调用 handleTDGAPushMessage 传入消息。

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    [TalkingDataGA onStart:@"APP_ID" withChannelId:nil];
    if (![TalkingDataGA handleTDGAPushMessage:launchOptions]) {
        // other code
    }
    // other code
}

- (void)application:(UIApplication *)application
didReceiveRemoteNotification:(NSDictionary *)userInfo
{
    if (![TalkingDataGA handleTDGAPushMessage:userInfo]) {
        // other code
    }
}
```

五、 集成检查列表（CHECKLIST）

- 确保 SDK 被正确初始化：Android 平台在 activity 的 onCreate 方法中调用了 TalkingDataGA.init；iOS 平台调用了 TDCCTalkingDataGA::onStart；——必须
- 确保调用了 TDCCAccount::setAccount。——必须

  必要时请在玩家注册、登录、切换区服、老玩家每次打开游戏时都进行调用，以确保玩家账户稳定，数据精确。
- 如果您有添加对玩家充值的追踪，确保不仅只发送了 onChargeSuccess，在成功前必须发送对应订单的 onChargeRequest。确保每笔订单的 orderID 是唯一的。
- 进行数据测试时，请确保网络畅通。
- Android 平台下如果对游戏代码进行了混淆 ,需要在 Proguard 混淆配置文件里添加一行，来保证统计 SDK 不会被修改：`-keep class com.tendcloud.tenddata.**{*;}`

六、 FAQ

1. 我需要跟踪收入，但是调用接口时我不知道 orderID，怎么办？

答：orderID 您可以在每次订单发起时自己构造一个，保证其唯一性即可，您需要自己管理此 orderID，必要时交付给游戏服务器保存。

2. 为何安装了好几次游戏，设备激活量没有变化？

答：同一台设备反复卸载和安装游戏不会记录多次设备激活；如果是在多个设备上打开的游戏，请稍后几分钟刷新一下页面，查看数据是否变更。

3. TalkingGame 如何判定一台 iOS 设备呢？

答：TalkingGame 首先对操作系统版本进行校验，低于 iOS7 的系统中会通过获取设备 MAC 地址（en0）来确定唯一设备；而 iOS7 和以后的设备已经不可在获取这些信息，我们通过获取 IDFA 和 IDFV 来判定设备，获取不到时会使用 UUID 策略生成 ID。

4. 为何测试渠道包时，渠道数据报表中没有任何数据？

答：请确认是否是在同一台设备上在卸载和安装渠道包进行测试，每台设备只被记录到初装渠道，更换渠道包不会产生新的激活量。测试每个渠道包时请使用一台没测试过的设备进行。

5. 为何测试了多个设备上玩游戏，玩家数没有增加？

答：您是否是在多个设备上使用了相同的账户进行了登录呢，那么激活会记录多个，但是玩家数不会增加，使用新的账户进行游戏玩家数才会变化。

6. 为何游戏玩家账户数比设备激活量还高呢？

答：如果玩家在一台设备上更换多个账户登录，就会产生此种情况。如果游戏存在刷小号情况，此现象会更显著。

7. 为何我的玩家账户数高于设备量了，注册转化率却不到 100%呢？

答：注册转化率是衡量所有已激活设备里有多少比例已经成功注册了游戏账户，如果有些设备上用了多个账户做登录，但仍然存在还没有账户注册的激活设备就会存在此种情况。

8. 为何测试收入数据时，有付费次数和人数，但是收入金额是 0 呢？

答：请检查追踪收入当中是否只调用了 success，而没有调用 request 呢。

9. 我的游戏中有多种虚拟币，如何集成呢？

答：TalkingGame 目前只支持记录一种虚拟币，如果您存在多币种，建议按照自身的折算关系将另一种币种折算为首选虚拟币来进行记录。

10. 游戏的每个账户下支持多个角色，如何集成？

答：您可以使用角色 ID 在作为 accountID 来进行集成，但是请注意，如此集成时，总的玩家账户数量会比实际值高。

技术支持

如果您在集成和阅读数据时遇到任何问题，请及时与我们取得联系：

技术支持邮箱：support@tendcloud.com

企业 QQ：4008701230

热线：4008701230