1. 下载SDK

下载iOS SDK压缩包，压缩包中 `TTTracker.framework` 文件为您App需要嵌入的SDK，压缩包中 `TrackerExternalExample` 是我们提供SDK接入示例工程。注意，本文提到的 `TTTracker.framework` 指的是您实际下载的SDK版本。

2. 导入SDK

解压下载下来的SDK，可以看到 `TTTracker.framework` ， `TrackerExternalExample` 示例程序，接入文档。这里以 `TrackerExternalExample` 示例程序导入SDK为例：

1. 将 `TTTracker.framework` Move到示例程序项目lib文件内。
2. 如弹出 `Choose options for adding these files:` 弹窗必须勾选 `Destination: Copy items if needed` 确保 `TTTracker.framework` 被引入到你的工程目录下。
3. 在具体代码引入的地方将对应接口import进来。目前SDK最低支持系统版本号为iOS 7.
   `#import <TTTracker/TTTracker.h>`

3. SDK配置

SDK配置是可选的，根据需求确定是否需要配置。配置必须发生在调用 `-startWithAppID: channel: appName:` 方法初始化SDK前

- 使用方可以配置上报的日志是否需要加密，设置登录用户的user*id，或者user*unique*id*等。

```
[[TTTracker sharedInstance] setConfigParamsBlock:^(void) {
    NSMutableDictionary *params = [NSMutableDictionary dictionary];
    [params setValue:@"123455" forKey:@"user"uniqueid"];
    [params setValue:@(YES) forKey:@"need"encrypt"];

    return [params copy];
}];
```

- SDK默认上传全渠道数据，使用方可以配置为仅上传头条渠道的数据

```
[[TTTracker sharedInstance] setSessionEnable:NO];
//是否开启session，YES开启，NO关闭，default= YES
@property (nonatomic, assign) BOOL sessionEnable;
```

- 使用方可以自定义header中字段扩展原来header中的字段，扩展的header字段与其他字段不在同一层级，统一封装在custom结构中（只是扩展，不可以覆盖，有覆盖需求的话看下面的接口）

```
[[TTTracker sharedInstance] setCustomHeaderBlock:^(void) {
        return @{@"region":@"cn"};
    }];
```

4. SDK初始化

找到您的App的入口（AppDelegate.m）并导入头文件，并在didFinishLaunchingWithOptions方法中调用[TTTracker startWithAppID:@”10008” channel:@”local*test”] 方法，这样就完成了初始化SDK的工作。*

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    //appID和channel分别表示app唯一标示（头条数据仓库组分配）和App发布的渠道名（建议内测版用localtest，正式版用App Store，灰度版用发布的渠道名，如pp）
   [TTTracker startWithAppID:@"10008" channel:@"localtest" appName:@"test"];
    return YES;
}
```

5.上报行为数据

1. 游戏预埋事件上报:商业化SDK对外提供了一些预先定义好的事件，接入方可以恰当的时机直接调用相关方法上报对应事件（注册register事件必传；支付purchase事件必传，其中仅isSuccess和currency_amount必传，currency_amount不能为0），游戏预埋事件API在TTTeacker+Game.h文件中
   注册事件上报示例：

   ```
   [TTTracker registerEventByMethod:@"weixin"
                           isSuccess:YES];
                           /**
                           Method：注册方式
                           isSuccess: 是否成功
                           */
   ```

   支付事件上报示例：

```
   [TTTracker purchaseEventWithContentType:@"mingwen"
                                 contentName:@"zuanshi"
                                   contentID:@"2345556"
                                contentNumber:100
                               paymentChannel:@"weixin"
                                     currency:@"rmb"
                              currency_amount:80000
                                    isSuccess:YES];

                                     /**
                                ContentType：购买内容类型
                                contentName：购买内容
                                contentNumber：购买内容编号
                                paymentChannel：支付渠道
                                currency：货币类型
                                currency_amount：支付金额（必须上报，不能为0）
                                isSuccess：是否成功付费（必须上报）

                                */
```

升级事件上报示例：（如后续投放转化目标为 升级则创建角色、升级事件必须上报。）

```
(void)updateLevelEventWithLevel:(NSUInteger)level;

/**
level :等级
 */
```

2. 自定义事件上报 在App内发生转化行为时，可以调用下面的代码上报行为数据。

```
//第一个参数为在头条数据后台注册的事件名称，第二个参数为本次事件需要上传的参数
[TTTracker eventV3:@"toutiao" params:@{@"is_log_in":@(1)}];
```

游戏行业建议上报的自定义埋点事件为：创建角色、升级。如后续投放转化目标为创建角色、升级则创建角色、必须上报。请按如下埋点设计进行上报

事件说明	event	param	属性描述	属性类型
创建角色	create_gamerole	gamerole_id	角色id	string

注：自定义埋点事件设计原则如下：
(1) event命名仅支持字母, 数字和下划线，不要使用launch、terminal等系统内置事件名，服务端event命名

尽量以server开头
(2) 目前param仅接受 int与string类型，最好不要上报boolean与float
(3) 强调1：不要在param中再嵌套字典，即其value不接受dic类型
(4) 强调2：param暂不推荐数组类型，我们一般情况下会整个当string处理
(5) 数值类型的参数param最好不要打成string，影响后续的参数计算与标注显示
(6) 不建议param用“”或“ ”代表实际含义，建议用“be_null”，同时不建议用到 -1，其被系统占用了

6.设置用户唯一标识（可选）

UserUniqueID作为用户的唯一的标识，传入此值可以以用户为单位进行统计，如果不传此id，则以设备为单位进行统计（如果以设备为单位统计，当用户换设备之后，数据统计平台就认为是两个统计单位）。通常接入方可以以业务方平台登陆的userId作为UserUniqueId传入（注，当用户登出的时候请把UserUniqueId置为nil，否则SDK不知道用户已经登出，仍然将统计结果计算在该用户身上）。

用户登入时需调用-setCurrentUserUniqueID:接口传入当前的用户的user_unique_id

```
[[TTInstallIDManager sharedInstance] setCurrentUserUniqueID:[UserAccount userID]];
```

用户退出登入时需要调用-setCurrentUserUniqueID:接口并传入nil

```
[[TTInstallIDManager sharedInstance] setCurrentUserUniqueID:nil];
```