IOS SDK接入说明文档

# 快速接指引

## 1. 获取SDK

### 方式一：手动引入

手动引入，下载SDK

图形用户界面, 应用程序

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

选择拷贝需要的framework到您的应用目录下，**放置目录无具体要求**，

放置举例：

可以在Xcode中需要添加的Target中选择 ”Build Phases”->“Link Binary With Libraries”->“Add”->“Add Other”→选择framework目录。

A screenshot of a computer

Description automatically generated with medium confidence

在“Build Setting”->”linking”->”Other linker flags”里加入-ObjC标志

A screenshot of a computer

Description automatically generated with medium confidence

### 方式二：cocoapods引入

使用cocoapods的方式，需要在Podfile文件添加腾讯的podspec源，并pod 依赖灯塔SDK，参考如下：

source 'https://github.com/CocoaPods/Specs.git'

platform :ios, '9.0'

target 'Demo' do

use\_frameworks!

pod 'tgp\_beacon', '~> 2.1.2'

end

## 2. 初始化SDK

引入头文件

#import <BeaconAPI\_Base/BeaconReport.h>

初始化SDK

在**AppDelegate** 的**- application: didFinishLaunchingWithOptions:** 中添加初始化代码：

- (BOOL)application:(UIApplication \*)application

didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions {

// 配置渠道id，可用于分析用户来源

BeaconReport.sharedInstance.channelId = @"渠道id";

// 设置app版本

[BeaconReport.sharedInstance.getCommonParams setAppVersion: @"app版本"];

[BeaconReport.sharedInstance initOstar]; // startWithAppkey 之前必须调用

BeaconReportConfig \*reportConfig = [BeaconReportConfig new];

// 私有化部署 初始化必须配置上报地址！！！否则走默认的saas上报地址

reportConfig.uploadURL = @"上报的url地址";

// 填写从平台创建应用时生成的appkey

[BeaconReport.sharedInstance startWithAppkey:"申请的Appkey" config:reportConfig];

// 是否开启Socket上报，需要设置为NO

[BeaconReport.sharedInstance setSocketOnOff:NO];

return YES;

}

初始化时代码文件位置，如下指引：

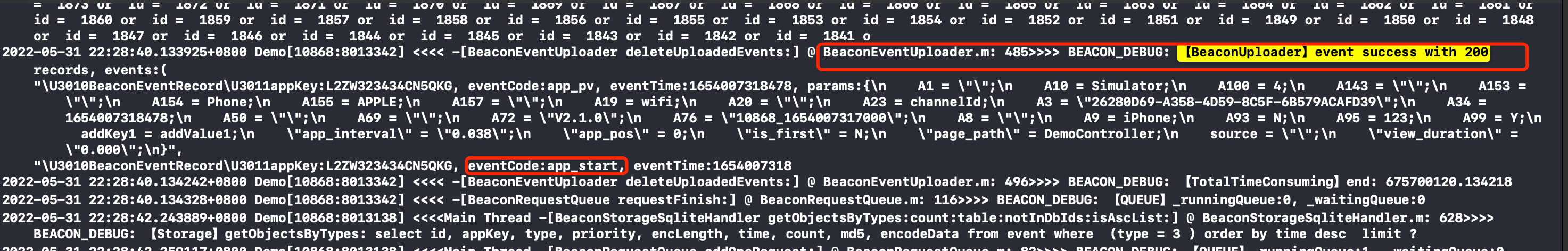
A screenshot of a computer

Description automatically generated

**至此SDK已完成初始化，会开始自动采集预置事件和预置参数，向平台上报数据。**

## 3. 判断上报状态

可以在debug模式下，查看日志，出现【BeaconUploader】event success with 200表示上报成功



# 平台操作指引

## 查看AppKey

AppKey需在SDK初始化时传入，决定了数据上报到哪个数据源中。

渠道一：【数据管家】  
Graphical user interface, application

Description automatically generated

渠道二：【应用管理】  
图形用户界面, 文本, 应用程序

描述已自动生成

## 登记事件信息

当预置事件无法满足业务需求时，可以在平台内登记自定义事件，再调用SDK API上报自定义事件。平台操作如下：

1. 进入到数据源

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

1. 登记事件（创建登记事件或查看登记事件）

Graphical user interface, application

Description automatically generated

图形用户界面, 应用程序

描述已自动生成

## 3. 查看上报日志

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面, 应用程序, Word

描述已自动生成

在平台数据资源中预览到上报事件记录，即可判断接入成功。

# 自定义上报

// 参数

NSDictionary \*params = @{@"button\_name": @"report\_button1"};

// 事件code

NSString \*normal\_event\_code\_test = @"testDemoButtonClick";

// 上报实时事件

BeaconEvent \*realTimeEvent = [BeaconEvent realTimeEventWithCode:normal\_event\_code\_test params:params];

[BeaconReport.sharedInstance reportEvent:realTimeEvent];

// 上报普通事件

BeaconEvent \*noralEvent = [BeaconEvent normalEventWithCode:normal\_event\_code\_test params:params];

[BeaconReport.sharedInstance reportEvent:normalEvent];

# SDK API汇总

## 1 初始化接口进阶

设置上报配置：BeaconReportConfig(**以下配置都可以不设置，不设置会走默认配置**)

BeaconReportConfig \*config = [BeaconReportConfig new];

// 本地数据库的最大容量（超过限额不予存储），默认10000条，保护区间是100～100000条，云端优先级高于本地设置

config.maxDBCount = 10000;

// 实时事件上报的轮询间隔，默认2s，允许区间是[0.1,20]s

config.realTimeEventPollingInterval = 2;

// 普通事件上报的轮询间隔，默认5s，允许区间是[1,50]s

config.normalEventPollingInterval = 5;

## 2 设置全局ID

// 设置用户唯一标识符，用以通过userId标识和分类异常用户信息

[BeaconReport.sharedInstance setUserId: @"userId" forAppKey:@"appKey"];

**注意：**

1. 如果后续想要使用画像洞察分析的相关功能，需要在此字段传入用户的**设备标识符**或**手机号**。（推荐传入手机号）
2. 设备标识符支持Android的**IMEI**和iOS的**IDFA**。
3. 设备标识符或手机号可以MD5加密后再传入（32位小写）。

## 3 设置app版本

// 设置app版本

[BeaconReport.sharedInstance.getCommonParams setAppVersion: @"app版本"];

## 4 添加额外上报参数

// 添加额外上报参数

[BeaconReport.sharedInstance setAdditionalParams:@{@"addKey1" : @"addValue1" , @"app\_pos" : @(0)} forAppKey:self.appKey];

## 5 App 打通 H5

集成了 Web JS SDK 的 H5 页面，在嵌入到 App 后，H5 内的事件可以通过 App 进行发送，事件发送前会添加上 App 采集到的预置属性。该功能默认是关闭状态，如果需要开启，需要在 App 和 H5 端同时进行开启。

// 在嵌入WKWebView的页面中，创建JsReport对象

JsReport \*jsReport = [JsReport new];

// 开启内嵌H5通过App上报的通路

[jsReport enableBridge:wkWebView];

// 关闭内嵌H5通过App上报的通路

[jsReport disableBridge];

## 6 停止上报、恢复上报

/// 停止上报, immediately为YES则会马上中断正在进行的任务，NO则会等待任务完成后再停止轮询

/// 停止上报

[BeaconReport.sharedInstance stopReport:YES];

/// 立即停止上报

[BeaconReport.sharedInstance stopReport:NO];

//恢复上报

[BeaconReport.sharedInstance resumeReport];

## 7 其他设置

// 设置本地调试时控制台输出的日志级别：1 fetal, 2 error, 3 warn, 4 info, debug, 5 debug, 10 all, 默认为0，不打印日志

// 线上正式环境，必须设置为0关闭此日志

BeaconReport.sharedInstance.logLevel = 0;

## 8 上报失败重试策略

上报失败后，上报记录会先存储在APP缓存中（存储条数受maxDBCount参数控制），并开启重试请求策略，请求的时间间隔要逐渐增加。

上报间隔与上报失败次数的关系为： 上报间隔 = 初始上报间隔 \* 2^(上报失败次数)。计时达到上报间隔时间，即尝试进行下次上报。

上报成功后，上报间隔恢复起始默认值; 如果客户设置了上报间隔，则恢复成其设置值。

# 本地启动demo操作流程

## 1. demo下载地址

Demo代码地址：<https://github.com/tgrowing/tgp_beacon_iOS>

## 2. 运行demo

进入到Demo目录下  
Graphical user interface, text, website

Description automatically generated  
在Demo目录下，执行pod install --repo-update,安装完成如下图展示  
Text

Description automatically generated  
双击（打开）生成的Demo.xcworkspace，运行即可  
Graphical user interface, text, application

Description automatically generated

## 3. 参数填写示例 Graphical user interface, application Description automatically generated

## 4. 查看上报

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面, 应用程序, Word

描述已自动生成

# 附录

预置事件表

|  |  |
| --- | --- |
| 事件code | 事件显示名 |
| app\_start | APP启动 |
| app\_init | SDK初始化 |
| app\_pv | 页面访问 |
| app\_unpv | 页面关闭 |
| app\_end | APP退出 |

**注：**预置事件由SDK自动采集，无需再调用主动上报的埋点代码

预置参数表

|  |  |
| --- | --- |
| 参数KEY | 参数显示名 |
| event\_code | 事件code |
| event\_time | 上报时间 |
| uin | 用户标识码 |
| uid | 自定义 UID |
| country | 国家 |
| province | 省份 |
| city | 城市 |
| sdk\_version | SDK 版本 |
| channel | 自定义渠道 |
| brand | 品牌 |
| model | 机型 |
| platform | 终端平台 |
| os\_version | 操作系统版本号 |
| os | 操作系统类型 |
| operator | 运营商 |
| product\_id | 应用 id |
| network\_type | 网络类型 |
| view\_duration | 页面访问时长 |
| path\_parameter | 页面参数 |
| page\_path | 页面路径 |
| is\_first | 是否新用户 |
| source | 跳转来源 |
| screen\_resolution | 屏幕分辨率 |
| app\_version | APP版本 |
| app\_interval | APP访问时长 |

**注：**

* 所有预置参数中除了uid、channel、app\_version需要调用SDK接口传值，其余均由SDK自动采集上报。
* 预置参数会附在所有事件上一起上报（包括预置事件和自定义事件）