Bugly iOS SDK 接入指南

一. SDK 集成

Bugly提供两种集成方式供iOS开发者选择：

- CocoaPods
- 手动集成

如果您是从 `Bugly 2.0` 以下版本升级过来的，请查看 `Bugly旧版本顺滑升级指引`

1.1 CocoaPods集成方式

命令行下执行 `pod search Bugly` ,如显示的 `Bugly` 版本不是最新的，则先执行 `pod repo update` 操作

在工程的Podfile里面添加以下代码：

```
pod 'Bugly'
```

保存并执行 `pod install` ,然后用后缀为 `.xcworkspace` 的文件打开工程。

关于 `CocoaPods` 的更多信息请查看CocoaPods官方网站。

1.2 手动集成方式

- 下载并解压iOS SDK
- 拖拽 `Bugly.framework` 文件到Xcode工程内(请勾选 `Copy items if needed` 选项)
- 添加依赖库
  - `SystemConfiguration.framework`
  - `Security.framework`
  - `libz.dylib`

二. 初始化SDK

2.1 导入头文件

在工程的 `AppDelegate.m` 文件导入头文件

```
#import <Bugly/Bugly.h>
```

如果是 `Swift` 工程，请在对应 `bridging-header.h` 中导入

2.2 初始化Bugly

在工程 `AppDelegate.m` 的 `application:didFinishLaunchingWithOptions:` 方法中初始化Bugly：

- **Objective-C**

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
    [Bugly startWithAppId:@"此处替换为你的AppId"];
    return YES;
}
```

- **Swift**

```
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [
    Bugly.startWithAppId("此处替换为你的AppId")
    return true
}
```

如果您需要上报 `iOS Watch2 App` 或 `iOS App Extension` 的异常，请参见Bugly iOS Extension SDK接入指南。

至此，Xcode工程集成Bugly已完成，接下来可以编译并运行你的Xcode工程。

三. 高级功能

Bugly SDK提供一系列自定义配置或接口供开发者调用，如:

- 文件配置初始化参数
- 自定义日志
- 界面追踪
- 卡顿监控

详细内容请查看iOS SDK高级设置。

Bugly iOS SDK高级功能使用指南

一. 文件配置初始化参数

Bugly支持读取 `Info.plist` 文件读取SDK初始化参数，可配置的参数如下：

```
- Appid
    - Key: BuglyAppIDString
    - Value: 字符串类型
- 渠道标识
    - Key: BuglyAppChannelString
    - Value: 字符串类型
- 版本信息
    - Key: BuglyAppVersionString
    - Value: 字符串类型
- 开启Debug信息显示
    - Key: BuglyDebugEnable
    - Value: BOOL类型
```

如下初始化方式，则会读取 `Info.plist` 内添加的key-value配置进行SDK初始化：

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
    // 读取Info.plist中的参数初始化SDK
    [Bugly startWithAppId:nil];
    return YES;
}
```

二. 自定义日志

SDK提供自定义日志打印接口，用于记录一些关键的业务调试信息，可以更全面地反应App发生崩溃或异常时的上下文环境。

使用方式与NSLog一致，可定义日志级别以便过滤，日志接口宏定义如下：

```
BLYLogError(fmt, ...)
BLYLogWarn(fmt, ...)
BLYLogInfo(fmt, ...)
BLYLogDebug(fmt, ...)
BLYLogVerbose(fmt, ...)

void BLYLog(BuglyLogLevel level, NSString *format, ...);
void BLYLogv(BuglyLogLevel level, NSString *format, va_list args);
```

三. 接口声明

1. 功能配置

```objc
/**
 *  SDK Debug 信息开关，默认关闭
 */
@property (nonatomic, assign) BOOL debugMode;

/**
 *  设置自定义渠道标识
 */
@property (nonatomic, copy) NSString *channel;

/**
 *  设置自定义版本号
 */
@property (nonatomic, copy) NSString *version;

/**
 *  设置自定义设备唯一标识
 */
@property (nonatomic, copy) NSString *deviceId;

/**
 *  卡顿监控开关，默认关闭
 */
@property (nonatomic, assign) BOOL blockMonitorEnable;

/**
 *  卡顿监控判断间隔，单位为秒
 */
@property (nonatomic, assign) NSTimeInterval blockMonitorTimeout;

/**
 *  ATS开关，默认开启。如果关闭，SDK的网络请求不会通过HTTPS发送
 */
@property (nonatomic) BOOL appTransportSecurityEnable;

/**
 *  进程内还原开关，默认开启。
 */
@property (nonatomic) BOOL symbolicateInProcessEnable;

/**
 *  非正常退出事件记录开关，默认关闭
 */
@property (nonatomic) BOOL unexpectedTerminatingDetectionEnable;

/**
 *  页面信息记录开关，默认开启
```

```
 */
@property (nonatomic) BOOL viewControllerTrackingEnable;

/**
 *  SDK回调
 */
@property (nonatomic, assign) id<BuglyDelegate> delegate;

/**
 * 控制自定义日志上报，默认值为BuglyLogLevelWarn，只上报Warn、Error的日志。
 * 设置为BuglyLogLevelSilent可关闭日志上报。
 */
@property (nonatomic, assign) BuglyLogLevel reportLogLevel;
```

2. SDK回调

```
/**
 *  发生异常时回调
 *
 *  @param exception 异常信息
 *
 *  @return 返回需上报记录，随异常上报一起上报
 */
- (NSString *)attachmentForException:(NSException *)exception;
```

3. 功能接口

```
/**
 *  初始化Bugly,使用默认BuglyConfig
 *
 *  @param appId 注册Bugly分配的应用唯一标识，如果appId为空，则读取Info.plist中的参数配置
 */
+ (void)startWithAppId:(nullable NSString *)appId;

/**
 *  使用指定配置初始化Bugly
 *
 *  @param appId 注册Bugly分配的应用唯一标识
 *  @param config 传入配置的 BuglyConfig
 */
+ (void)startWithAppId:(nullable NSString *)appId
                config:(nullable BuglyConfig *)config;

/**
 *  设置用户标识
```

```
 *
 *  @param userId 用户标识
 */
+ (void)setUserIdentifier:(nonnull NSString *)userId;

/**
 *  更新应用版本信息
 *
 *  @param version 应用版本信息
 */
+ (void)updateAppVersion:(NSString *)version;

/**
 *  设置关键数据，随崩溃信息上报
 *
 *  @param value
 *  @param key
 */
+ (void)setUserValue:(nonnull NSString *)value
              forKey:(nonnull NSString *)key;

/**
 *  获取关键数据
 *
 *  @return 关键数据
 */
+ (nullable NSDictionary *)allUserValues;

/**
 *  设置标签
 *
 *  @param tag 标签ID，可在网站生成
 */
+ (void)setTag:(NSUInteger)tag;

/**
 *  获取当前设置标签
 *
 *  @return 当前标签ID
 */
+ (NSUInteger)currentTag;

/**
 *  上报自定义异常
 *
 *  @param exception 异常信息
 */
+ (void)reportException:(nonnull NSException *)exception;
```

```objc
/**
 *  上报错误
 *
 *  @param error 错误信息
 */
+ (void)reportError:(NSError *)error;

/**
 *  SDK 版本信息
 *
 *  @return
 */
+ (nonnull NSString *)sdkVersion;

/**
 *  获取设备ID
 *
 *  @return 设备ID
 */
+ (nonnull NSString *)deviceId;
```