Bugly Unity Plugin 接入指南

一、概述

Bugly Unity Plugin是专为**Unity**平台的移动终端(iOS/Android)游戏APP而开发的崩溃/异常捕获组件，它能够自动捕获C#脚本未处理的异常和原生代码(Objective-C、Java、C/C++等)未处理的异常，并提供实时、多维度的异常问题分析服务。

> **准备工作**
>
> 如果你之前已经在**Bugly**或**腾讯移动开放平台**注册了应用，并获取到**App ID**，可以继续使用它。 如果你尚未注册应用，可以通过QQ登录**Bugly**网站，点击"用户名"，选择"我的App"，点击注册新App，填写完应用基本信息完成注册，即可获得**App ID**。
>
> **注意：** 同一产品iOS和Android平台的App**不能公用同一个App ID**

二、通用部分集成步骤

1. 下载并导入Bugly Unity Plugin到Unity项目工程

- 下载最新版本Bugly Unity Plugin，双击**.unitypackage**文件，导入Plugin的相关文件到您的Unity工程中。

  > 下载包目录结构说明
  >
  > - bugly_plugin_*.unitypackage - Bugly Unity Plugin包, 提供C#异常捕获功能及原生SDK接口封装
  > - BuglySDK - Bugly原生SDK, Plugin包依赖的原生SDK
  > - BuglyUnitySample - Unity工程示例

  双击**bugly_plugin_*.unitypackage**文件，根据平台需求导入文件到Unity工程(如果你正在使用旧版本Plugin包，请务必先删除旧版本相关的文件)

  > bugly_plugin_*.unitypackage目录结构说明
  >
  > - Assets/Plugins/BuglyPlugins - Plugin脚本
  > - Assets/Plugins/BuglyPlugins/Android/libs - Android平台依赖的原生SDK(.jar)及NDK组件(.so)
  > - Assets/Bugly.framework - iOS平台依赖的原生SDK静态库(默认使用libc++编译静态库, 如需使用libstdc++编译的静态库,可以使用BuglySDK/iOS/libstdc++/Bugly.framework替换)

> 注意:
>
> 1. 集成**Bugly Unity Plugin**之后，还需集成对应的iOS或Android平台的SDK组件
> 2. iOS的SDK组件可以在导出的Xcode工程中集成并修改配置(具体可以参考iOS SDK接入指南的工程配置章节)，或者使用XUPorter等插件自动集成，Unity 5.x将会自动集成工程目录的iOS静态库，但仍需在Xcode工程中修改必要的工程配置
> 3. Android的SDK组件可以在导出的Android工程中集成并修改配置(具体可以参考Android SDK接入指南)，或直接把组件的内容拷贝到工程Plugins/Android目录下，并修改AndroidManifest.xml的权限声明

2. 初始化Bugly

- 选择第一个或主场景(*Scene*)，在任意脚本文件(建议选择较早加载的脚本)中调用如下代码进行初始化。

```
    // 开启SDK的日志打印，发布版本请务必关闭
    BuglyAgent.ConfigDebugMode (true);
    // 注册日志回调，替换使用 'Application.RegisterLogCallback(Application.LogCallback)'注册日志回调的方式
    // BuglyAgent.RegisterLogCallback (CallbackDelegate.Instance.OnApplicationLogCallbackHandler);

    #if UNITY_IPHONE || UNITY_IOS
        BuglyAgent.InitWithAppId ("Your App ID");
    #elif UNITY_ANDROID
        BuglyAgent.InitWithAppId ("Your App ID");
    #endif

    // 如果你确认已在对应的iOS工程或Android工程中初始化SDK，那么在脚本中只需启动C#异常捕获上报功能即可
    BuglyAgent.EnableExceptionHandler ();
```

三、iOS部分集成步骤

`不发布到ios平台可略过此部分`

1. 在Unity中修改项目的编译设置(Build Settings)

- 按下 `Ctrl+Shift+B` 打开*Build Settings*面板，点击*Player Settings ...*，切换到*Setting for iOS*选项卡，选择*Other Settings*栏，修改**Optimization**配置项**Script Call Optimization**的值为**Slow and Safe**

2. 修改导出的Xcode工程的编译配置

> 此部分的配置你可以参考iOS SDK接入指南

- 切换到*Build Phases*选项卡，在Link Binary With Libraries栏目下添加如下依赖项：

> - **libz.dylib** - 用于对上报数据进行压缩
> - **Security.framework** - 用于存储keychain
> - **SystemConfiguration.framework** - 用于读取异常发生时的系统信息
> - **JavaScriptCore.framework** - 设置为Optional
> - **libc++.dylib** - 用于支持libc++编译的项目，如果你的项目使用**libstdc++**编译，请更新替换使用libstdc++编译的SDK
>
> *注意：*
>
> 1) 如果Unity项目开启IL2CPP编译以支持ARM64位，Xcode工程默认C++编译配置为libc++，请根据Xcode工程的C++ Standard Library配置选择对应的Bugly.framework集成
>
> 2) 如果项目已经添加过这些依赖项，不用重复添加

至此、Unity项目的iOS工程配置完成。你可以在Unity中触发C#的异常验证崩溃上报功能。

四、Android部分集成步骤

`不发布到Android平台可略过此部分`

修改Android工程的配置文件AndroidManifest.xml

> 此部分的配置你可以参考Android SDK接入指南
>
> - 修改导出的Android工程的AndroidManifest.xml文件中的权限声明，添加如下权限：

```
    <!-- 网络通信-->
    <uses-permission android:name= "android.permission.INTERNET" />
    <!-- 获取网络状态 -->
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" /
>
    <!-- 获取MAC地址-->
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <!-- 获取设备信息 -->
    <uses-permission android:name= "android.permission.READ_PHONE_STATE" />
    <!-- 可选的权限:  -->
    <!-- 获取logcat日志 -->
    <uses-permission android:name="android.permission.READ_LOGS" />
```

> 注意：如果权限声明已经添加，不用重复添加

至此、Unity项目的Android工程配置完成。你可以在Unity项目中触发C#的异常进行测试验证。

五、API列表

- **BuglyAgent.InitWithAppId(string)**

  > 初始化Bugly，传入Bugly网站注册获得的App ID。
  >
  > 启用native code(Obj-C、C/C++、Java)异常、C#异常捕获上报，如果你已经在相应的iOS或Android工程中初始化Bugly，那么你只需调用 `BuglyAgent.EnableExceptionHandler` 开启C#异常捕获上报即可。

- **BuglyAgent.EnableExceptionHandler()**

  > 启动C#异常日志捕获上报，默认自动上报级别LogError，那么LogError、LogException的异常日志都会自动捕获上报。
  >
  > 日志级别定义参考LogSeverity : {LogDebug、LogWarning、LogAssert、LogError、LogException}

- **BuglyAgent.RegisterLogCallback(BuglyAgent.LogCallbackDelegate)**

  > 注册LogCallbackDelegate回调方法， 处理系统的日志。
  >
  > 如果你的应用需要调用Application.RegisterLogCallback(LogCallback)等注册日志回调，你可以使用此方法进行替换。

- **BuglyAgent.ConfigAutoReportLogLevel(LogSeverity)**

  > 设置自动上报日志信息的级别，默认LogError，则>=LogError的日志都会自动捕获上报。
  >
  > 日志级别的定义有LogDebug、LogWarning、LogAssert、LogError、LogException等

- **BuglyAgent.ReportException (System.Exception, string)**

  > 上报已捕获C#异常，输入参数异常对象，附加描述信息

- **BuglyAgent.ReportException (string, string, string)**

  > 上报自定义错误信息，输入错误名称、错误原因、错误堆栈等信息

- **BuglyAgent.SetUserId (string)**

  > 设置用户标识，如果不设置，默认为Unknown。

> 在初始化之后调用

- **BuglyAgent.PrintLog (LogSeverity, string)**

  > 调用原生SDK的接口打印日志，打印的日志信息将跟错误信息一起上报
  >
  > 在初始化之后调用

- **BuglyAgent.ConfigDebugMode (bool)**

  > 开启本地调试日志打印，默认关闭
  >
  > 注意：在发布版本中请务必关闭调试日志打印功能

- **BuglyAgent.ConfigDefault (string, string, string, long)**

  > 修改应用默认配置信息：渠道号、版本、用户标识等。
  >
  > 在初始化之前调用
  >
  > 渠道号默认值为空，
  >
  > 版本默认值
  >
  > - Android应用默认读取AndroidManifest.xml中的android:versionName
  > - iOS应用默认读取Info.plist文件中CFBundleShortVersionString和CFBundleVersion，拼接为CFBundleShortVersionString(CFBundleVersion)格式，例如1.0.1(10)
  >
  > 用户标识默认值10000

- **BuglyAgent.ConfigAutoQuitApplication (bool)**

  > 配置是否在捕获上报C#异常信息后就立即退出应用，避免后续产生更多非预期的C#的异常。
  >
  > 在初始化之前调用

六、FAQ

1. 初始化SDK后，为什么仍然无法捕获上报C#异常？

   > 答: 如果无法捕获上报C#异常，可以可以按照下面方式进行排查：
   >
   > 1. 检查是否有其他存在注册 `Application.RegisterLogCallback(LogCallback)` 的逻辑，由于系统默认的 `LogCallback` 是单播实现， 所以只能维持一个回调实例，你可以调用 `BuglyAgent.RegisterLogCallback(BuglyAgent.LogCallbackDelegate)` 方法来替代日志回调的注册。

> 2. 检查对应平台的SDK组件是否已集成到项目。

2. 为什么发生C#异常后，应用直接崩溃?

> 答: 如果遇到此场景，可以把脚本中 `InitWithAppId` 注释，保留 `EnableExceptionHandler` 方法调用，并在对应的Android或iOS工程中初始化SDK的组件。