

JCSDK 对接文档说明

Version: 1.0.0

目录

一、SDK 简介：	2
1.1、支持广告类型：	2
1.2、版本记录：	2
二、SDK 接入配置（带 unity 桥接和配置文件）：	3
2.1 info.plist 配置：	3
2.2 build setting 配置：	3
2.3 导入相关 SDK：	4
2.4 所需系统支持库：	4
2.5 JCiOSConfig.plist 文件说明：	5
以下版本有改动时，请参照版本 sdk 内部带的 plist.....	5
三、unity 接入配置和 API 说明：	6
3.1 JC_unityAdApi.h 接口说明：	6
3.2 JC_unityCallBackApi.h 接口说明：	7
四、广告接口 API 和回调使用.....	10
4.1 初始化 和 splash 广告 api 说明：	10
4.2 banner 广告 api 说明：	11

4.3 插屏广告 api 说明:	12
4.4 激励视频广告 api 说明:	13
4.5 回调示例.....	14
五 、相关报错.....	16
4.1 启动时崩溃.....	16
4.2 App 打包失败/打包提交失败.....	17
六、 iOS14 支持说明	20

一 、SDK 简介：

JCSDK 是 MS 公司提供的一套广告类型的 SDK，内部集成了各大广告商的广告 SDK 和相关数据统计 SDK，便于平台之间对应用内广告的联合运营和数据分析。

1.1、支持广告类型：

开屏广告、banner 广告、激励视频广告、插屏广告、native 广告

1.2、版本记录：

请参阅 [版本记录](#)

二 、SDK 接入配置（带 unity 桥接和配置文件）：

以下是导出 Xcode 所需的配置

但我们提供了桥接文件和配置文件，来自动集成一些配置，请查看参考使用：



2.1 info.plist 配置：

支持 http 网络配置、Google 相关参数配置：

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>
```

Google 所需参数配置：

```
<key>GADApplicationIdentifier</key>
<string>ca-app-pub-9488501426181082/7319780494</string>
<key>GADIsAdManagerApp</key>
<true/>
```

2.2 build setting 配置：

xcode 内 build setting 下配置：

bitcode 设置 NO,

Other linker flags 设置-ObjC

2.3 导入相关 SDK:

MS 平台支持的库和文件: [JCSDK](#)























JCSDK.framework 、JCiOSConfig.plist

第三方数据收集平台支持的库和文件: [DataCollection SDK](#)

第三方广告支持的库和文件: [ADThirdParty SDK](#)

2.4 所需系统支持库:

▼ Frameworks, Libraries, and Embedded Content

Name	Embed
 Accelerate.framework	Do Not Embed ⌵
 AdSupport.framework	Do Not Embed ⌵
 AVFoundation.framework	Do Not Embed ⌵
 CoreGraphics.framework	Do Not Embed ⌵
 CoreLocation.framework	Do Not Embed ⌵
 CoreMedia.framework	Do Not Embed ⌵
 CoreMotion.framework	Do Not Embed ⌵
 CoreTelephony.framework	Do Not Embed ⌵
 iAd.framework	Do Not Embed ⌵
 libbz2.tbd	
 libc++.tbd	
 libresolv.9.tbd	
 libsqlite3.tbd	
 libxml2.tbd	
 libz.tbd	
 MessageUI.framework	Do Not Embed ⌵
 SafariServices.framework	Do Not Embed ⌵
 Security.framework	Do Not Embed ⌵
 SystemConfiguration.framework	Do Not Embed ⌵
 UIKit.framework	Do Not Embed ⌵
 VideoToolbox.framework	Do Not Embed ⌵
 WebKit.framework	Do Not Embed ⌵
+ -	

AppTrackingTransparency.framework (iOS14 支持)

2.5 JCIOSConfig.plist 文件说明：

以下版本有改动时，请参照版本 sdk 内部带的 plist

KEY	Value explain
appid	和初始化 API 上所需的 appid 一样，二选一传入

channelid	和初始化 API 上所需的 channelId 一样，二选一传入
splashArealD	splash 广告位 ID
bannerArealD	banner 广告位 ID
interArealD	intersitial 广告位 ID
rewardVideoArealD	rewardVideo 广告位 ID
nativeArealD	native 广告位 ID
ReYunAppID	热云参数 appid（不可为空）
ReYunChannelID	热云参数 channelId（不可为空）
UmengAppID	友盟参数 appid（作为本地缓存参数，可为空但不建议为空）
ShuShuAppID	数数参数 appid（作为本地缓存参数，可为空但不建议为空）
TalkingDataAppID	talkingDataSDK 参数 appid（作为本地缓存参数，可为空但不建议为空）

三 、unity 接入配置和 API 说明：

3.1 JC_unityAdApi.h 接口说明：

```

/// Initialize sdk
/// @param isOpenLog Log Switch YES/NO
/// @param isShow 首次是否展示开屏 YES/NO
/// @param block 开屏关闭、失败、不展示、网络超时等回传，可视为初始化完成的一个回调 block

```

```
-(void)initJCSDKWithLog:(BOOL)isOpenLog  
isFirstShowSplash:(BOOL)isShow  
splashClose:(unityBlock)block;
```

V2.0.0 修改初始化接口:

```
-(void)initJCSDKWithUnityShow:(unityBlock)block;
```

/// idReady 插屏

```
bool isReadyIntersitial();
```

/// show 插屏

```
void showIntersitial();
```

/// isReady 激励视频

```
bool isReadyRewardVideo();
```

/// show 激励视频

```
void showRewardVideo();
```

/// show banner 广告

```
void showBannerView();
```

/// remove banner 广告

```
void removeBannerView();
```

3.2 JC_unityCallBackApi.h 接口说明:

/// 注册回调监听 , 请在建立广告回传桥接前调用

void RegistCallBacknotifition();

/// 用于开屏回调

/// @param failLoad load 失败

/// @param didShow 展示成功

/// @param didClick 点击

/// @param didClose 关闭

void splash_CallBack(ResultHandler failLoad,ResultHandler didShow, ResultHandler didClick, ResultHandler didClose);

/// 用于插屏回调

/// @param failLoad load 失败

/// @param didShow 展示成功

/// @param failToShow 展示失败

/// @param didClose 关闭

/// @param didClick 点击

/// @param failToPlayVideo 播放 video 失败

/// @param startPlayingVideo 开始播放 video

/// @param endPlayingVideo 播放 video 完成

void Intersitial_CallBack(ResultHandler failLoad,ResultHandler didShow, ResultHandler failToShow, ResultHandler didClose,ResultHandler didClick,ResultHandler failToPlayVideo, ResultHandler startPlayingVideo, ResultHandler endPlayingVideo);

/// 用于 banner 回调

/// @param failLoad load 失败


```
/// @param didShow 展示成功
/// @param didClick 点击
/// @param didAutoRefresh 自动刷新
/// @param tapCloseBtn 点击功能关闭按钮
/// @param failToAutoRefresh 自动刷新失败
void banner_CallBack(ResultHandler failLoad, ResultHandler
didShow, ResultHandler didClick, ResultHandler
didAutoRefresh, ResultHandler tapCloseBtn, ResultHandler
failToAutoRefresh);
```

```
/// 用于激励视频回调
/// @param failLoad load 失败
/// @param didRewardSuccess 奖励成功
/// @param didClose 关闭
/// @param didClick 点击
/// @param failToPlayVideo 播放失败
/// @param startPlayingVideo 开始播放
/// @param endPlayingVideo 播放完成
void rewardVideo_CallBack(ResultHandler
failLoad, ResultHandler didRewardSuccess, ResultHandler
didClose, ResultHandler didClick, ResultHandler
failToPlayVideo, ResultHandler startPlayingVideo,
ResultHandler endPlayingVideo);
```

```
/// 用于原生广告回调（暂时未开放 native 广告功能）
/// @param failLoad load 失败
/// @param didShow 展示成功
/// @param didClick 点击广告
/// @param startPlayingVideo 开始播放
```

```

/// @param endPlayingVideo 播放完成
/// @param tapCloseBtn 点击关闭功能按钮
/// @param enterFullScreenV 进入全屏 video（用于模版）
/// @param exitFullScreenV exit 全屏 video（用于模版）
void native_CallBack(ResultHandler failLoad,ResultHandler
didShow, ResultHandler didClick, ResultHandler
startPlayingVideo, ResultHandler
endPlayingVideo,ResultHandler tapCloseBtn,ResultHandler
enterFullScreenV,ResultHandler exitFullScreenV);

```

四、广告接口 **API** 和回调使用

4.1 初始化 和 splash 广告 api 说明：

头文件：

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

4.1.1、OC 接口，在进入程序代理中>window 加载之后被调用

```

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    [[ADViewShowApi getInstance]initJCSDKWithLog:YES];
    return YES;
}

```

4.1.2、如果你的应用是游戏应用，请在 UnityAppController.mm 中实现

为了在 unity 前展示 splash 广告，需要找到下面代码并替换，可以 unity 硬替换，也可转 xcode 工程后手动操作

```
330         _startUnityScheduled = true;
331         //[[self performSelector: @selector(startUnity:) withObject: application
332         //    afterDelay: 0];
333         [self performSelector: @selector(initSDKWithApplication:) withObject:
334             application afterDelay: 0];
335     }
336     _didResignActive = false;
337 }
338 -(void)initSDKWithApplication:(UIApplication*)application{
339     [[JC_unityAdApi getInstance]initJCSDKWithLog:YES isFirstShowSplash:NO
340         splashClose:^(BOOL isOk) {
341             [self performSelector: @selector(startUnity:) withObject: application
342                 afterDelay: 0];
343         }];
344 }
```



这里是展示完开屏，再去调用 startUnity

4.2 banner 广告 api 说明：

showbanner 内部原理 load - isReady - show ， 删除 banner 内部会自动 load banner 广告 ， 最好不要删除后直接 show ， load 有缓冲期

```
[DllImport("__Internal")]
static extern void showBannerView();

[DllImport("__Internal")]
static extern void removeBannerView();
```

```

public static void ShowBanner()
{
    showBannerView();
}

public static void removeBanner()
{
    removeBannerView();
}

```

4.3 插屏广告 api 说明：

建议 先判断 **value** 确定内部是否有广告位， 再 **show** 广告

```

[DllImport("__Internal")]
static extern bool isReadyIntersitial();

[DllImport("__Internal")]
static extern void showIntersitial();

public static bool IsInterReady()
{
    var value = isReadyIntersitial();
    Debug.Log("-----> IsInterReady:" + value);
    return value;
}

```

```
public static void ShowInter()
{
    showIntersitial();
}
```

4.4 激励视频广告 api 说明：

建议先判断 value 确定内部是否有广告位， 再 show 广告

```
[DllImport("__Internal")]
static extern bool isReadyRewardVideo();

[DllImport("__Internal")]
static extern void showRewardVideo();

public static bool IsRewardVReady()
{
    var value = isReadyRewardVideo();
    Debug.Log("-----> isReadyRewardV:" + value);
    return value;
}

public static void ShowRewardV()
{
    showRewardVideo();
}
```

4.5 回调示例

注：回调前先调用注册监听方法，建立连接

插屏回调示例：

```
[DllImport("__Internal")]
static extern void Intersitial_CallBack(IntPtr failLoad, IntPtr didShow,
IntPtr failToShow, IntPtr didClose, IntPtr didClick, IntPtr failToPlayVideo,
IntPtr startPlayingVideo, IntPtr endPlayingVideo);
```

//注册插屏回调

```
var handler11 = new ResultHandler(interFailLoad);
var fp11 = Marshal.GetFunctionPointerForDelegate(handler11);
var handler12 = new ResultHandler(interDidShow);
var fp12 = Marshal.GetFunctionPointerForDelegate(handler12);
var handler13 = new ResultHandler(interFailToShow);
var fp13 = Marshal.GetFunctionPointerForDelegate(handler13);
var handler14 = new ResultHandler(interDidClose);
var fp14 = Marshal.GetFunctionPointerForDelegate(handler14);
var handler15 = new ResultHandler(interDidClick);
var fp15 = Marshal.GetFunctionPointerForDelegate(handler15);
var handler16 = new ResultHandler(interFailToPlayVideo);
var fp16 = Marshal.GetFunctionPointerForDelegate(handler16);
var handler17 = new ResultHandler(interStartPlayingVideo);
var fp17 = Marshal.GetFunctionPointerForDelegate(handler17);
var handler18 = new ResultHandler(interEndPlayingVideo);
var fp18 = Marshal.GetFunctionPointerForDelegate(handler18);
Intersitial_CallBack(fp11, fp12, fp13, fp14, fp15, fp16, fp17, fp18);
```

```
//插屏回调

[MonoPInvokeCallback(typeof(ResultHandler))]
static void interEndPlayingVideo(string resultString)
{
    Debug.Log("插屏回调----->interEndPlayingVideo");
}

[MonoPInvokeCallback(typeof(ResultHandler))]
static void interStartPlayingVideo(string resultString)
{
    Debug.Log("插屏回调----->interStartPlayingVideo");
}

[MonoPInvokeCallback(typeof(ResultHandler))]
static void interFailToPlayVideo(string resultString)
{
    Debug.Log("插屏回调----->interFailToPlayVideo");
}

[MonoPInvokeCallback(typeof(ResultHandler))]
static void interDidClick(string resultString)
{
    Debug.Log("插屏回调----->interDidClick");
}

[MonoPInvokeCallback(typeof(ResultHandler))]
static void interDidClose(string resultString)
{
    Debug.Log("插屏回调----->interDidClose");
}

[MonoPInvokeCallback(typeof(ResultHandler))]
```

```
static void interFailToShow(string resultString)
{
    Debug.Log("插屏回调----->interFailToShow");
}

[MonoPInvokeCallback(typeof(ResultHandler))]
static void interDidShow(string resultString)
{
    Debug.Log("插屏回调----->interDidShow");
}

[MonoPInvokeCallback(typeof(ResultHandler))]
static void interFailLoad(string resultString)
{
    Debug.Log("插屏回调----->interFailLoad");
}
```

五 、相关报错

4.1 启动时崩溃

应用程序在启动时崩溃时因为您缺少了某些配置，我们这里给出几个例子

在 Build Settings 中的 Other Linker Flags 缺失-ObjC flag 配置项

解决方案 ： 添加-ObjC

引入 Admob 的 SDK 后，程序启动崩溃

解决方案：在 info.plist 中添加 Google 所需 key

```
<key>GADApplicationIdentifier</key>
```

```
<string>ca-app-pub-9488501426181082/7319780494</string>
```

```
<key>GADIsAdManagerApp</key> <true/>
```

引入快手 SDK 编译崩溃

解决方案：将 KSAdSDK.framework 的 Embed 修改为 Embed&Sign

展示广点通激励视频或者插屏广告时崩溃 -[AppDelegate window]

解决方案：在 AppDelegate.h 中添加 window 属性

4.2 App 打包失败/打包提交失败

快手 SDK 中包含 x86 二进制，苹果商店不支持模拟器资源

解决方案：

在 Build Phase -> New Run Scrip Phase

添加 new Run Script Phase 之后，会出现 Run Script，然后在里面添加一段脚本代码，如下

```
APP_PATH="${TARGET_BUILD_DIR}/${WRAPPER_NAME}"
```

```
# This script loops through the frameworks embedded in the application and
```

```
# removes unused architectures.
```

```
find "$APP_PATH" -name '*.framework' -type d | while read  
-r FRAMEWORK
```

```
do
```

```
    FRAMEWORK_EXECUTABLE_NAME=$(defaults read  
"$FRAMEWORK/Info.plist" CFBundleExecutable)
```

```
    FRAMEWORK_EXECUTABLE_PATH="$FRAMEWORK/$FR  
AMEWORK_EXECUTABLE_NAME"
```

```
    echo "Executable is  
$FRAMEWORK_EXECUTABLE_PATH"
```

```
    EXTRACTED_ARCHS=()
```

```
    for ARCH in $ARCHS
```

```
do

    echo "Extracting $ARCH from
$FRAMEWORK_EXECUTABLE_NAME"

    lipo -extract "$ARCH"
"$FRAMEWORK_EXECUTABLE_PATH" -o
"$FRAMEWORK_EXECUTABLE_PATH-$ARCH"

EXTRACTED_ARCHS+=("$FRAMEWORK_EXECUTABLE_P
ATH-$ARCH")

done

echo "Merging extracted architectures: ${ARCHS}"

lipo -o "$FRAMEWORK_EXECUTABLE_PATH-merged"
-create "${EXTRACTED_ARCHS[@]}"

rm "${EXTRACTED_ARCHS[@]}"

echo "Replacing original executable with thinned
version"
```

```
rm "$FRAMEWORK_EXECUTABLE_PATH"

mv "$FRAMEWORK_EXECUTABLE_PATH-merged"
"$FRAMEWORK_EXECUTABLE_PATH"

done
```

添加完脚本代码块后，勾选上 **Run script only when installing**，重新打包提交即可。

六、iOS14 支持说明

详情请见 [JCSDK iOS14](#) 说明文档