

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：	2
2.1、info.plist 配置：	2
2.2、build setting 配置：	3
2.3、iOS14 支持：（可选）	3
2.4、导入相关 SDK 库和文件	3
2.5 导入系统支持库	4
2.6、关于 JCiOSConfig.plist 内部参数说明	5
三、SDK 接入 API：	6
3.1、引入头文件：	6
3.2、初始化 SDK：	6
3.3、splash 广告 api：	7
3.4、banner 广告 api：	7
3.5、Intersitial 广告 api：	7
3.6、RewardView 广告 api：	8
3.7、native 广告 api：	8

3.8、广告回调 api:	9
3.9、关于欧盟地区展示 GDPR:	10
四 、相关报错.....	11

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：

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>

<key>GADApplicationIdentifier</key>

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

2.2、 build setting 配置：

bitcode 设置 NO,

Other linker flags 设置-ObjC

2.3、 iOS14 支持：（可选）

详细请见 [JCSDK iOS14](#) 支持文档说明

2.4、 导入相关 SDK 库和文件























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](#)

2.5 导入系统支持库

▼ 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

2.6、关于 JCIOSConfig.plist 内部参数说明

以下参数由 MS 平台提供，"xx"AreaID 在初始化 isOpenInBody 传入 YES 的情况下可以传空值，当初始化 isOpenInBody 传入 NO 时，表示不开启内部获取广告接口数据，"xx"AreaID 不能为空

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
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）
splashAreaID	splash ad space ID
bannerAreaID	banner ad space ID
interAreaID	intersitial ad space ID
rewardVideoAreaID	rewardVideo ad space ID
nativeAreaID	native ad space ID

三 、SDK 接入 API :

如果文档内 API 和 framework 内 API 有冲突, 请以 framework 内 API 为准。

3.1、引入头文件:

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

3.2、初始化 SDK:

开启日志开关 和 初始化 API 调用

```
- (BOOL)application:(UIApplication *)application  
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {  
    [JC_iOSAdApi setOpenPlatformLog:YES];  
    [JC_iOSAdApi  
     jcSDKInitConfigWithAppId:@"ff48e91e-043e-46fc-8097-eeed0  
a7f3281" channelId:@"IOS" isOpenInBody:YES block:^(BOOL  
isOk) {  
        //It is recommended that the load event for each type  
of ad be called here, as a pre-load process  
    }];  
    return YES;  
}
```

3.3、 splash 广告 api:

开屏请在 window 加载之后被调用

```
[JC_iOSAdApi loadSplashView];
```

3.4、 banner 广告 api:

推荐: 先调用 load 进行广告位“预热”, 展示之前判断 isReady 是否为 YES ,
请自行设计调用场景, api 最好不要连续, 以免未及时 load 到数据

```
[JC_iOSAdApi loadBannerConfig];  
  
BOOL isReady = [JC_iOSAdApi bannerIsReady]  
//con 传入当前控制器即可  
[JC_iOSAdApi showBannerViewWithCon:con];
```

3.5、 Intersitial 广告 api:

推荐调用顺序 load - isReady - show - isReady - show (sdk 内部采用了自动加载插屏资源功能, 外部使用只需要调用一次 load 接口)

```
[JC_iOSAdApi loadIntersitialConfig];  
  
BOOL isReady = [JC_iOSAdApi intersitialIsReady]
```

```
[JC_iOSAdApi showIntersitialView];
```

3.6、RewardView 广告 api:

推荐调用顺序 load - isReady - show - isReady - show (sdk 内部采用了自动加载激励视频资源功能，外部使用只需要调用一次 load 接口)

```
[JC_iOSAdApi loadRewardConfig];  
BOOL isReady = [JC_iOSAdApi rewardVIsReady]  
[JC_iOSAdApi showRewardView];
```

3.7、native 广告 api:

native 没有缓存池，每次使用调用 load ，判断 isReady 后再展示。

show 方法有返回值，返回根据 config 生成的广告 view

JCNativeConfig 是 native 展示广告位的配置类，请配置完整，否则可能导致加载视图异常，请将返回的 view 加载到需要显示的视图上


```
[JC_iOSAdApi
loadNativeConfigSize:CGSizeMake(CGRectGetWidth(self.view.
bounds), 350)]; //size: 请和展示的原生 view 大小相同，避免加载不
全

BOOL isReady = [JC_iOSAdApi nativeIsReady]

JCNativeConfig *config = [[JCNativeConfig alloc]init];
    config.ADFrame = CGRectMake(.0f, 200.0f,
CGRectGetWidth(self.view.bounds), 350.0f);

    config.mediaViewFrame = CGRectMake(0, 120.0f,
CGRectGetWidth(self.view.bounds), 350.0f - 120.0f);

    config.renderingViewClass = [[[CustomView
alloc]init] class];

    config.rootViewController = self;
    UIView *adview = [JC_iOSAdApi
showNativeConfigWithConfig:config];
// 添加 adview 到视图上
```

3.8、广告回调 api:

以下是 splash 广告的回调 api 使用示例，其他广告回调请自行使用

回调监听的 key，相关状态类型、回调参数说明请查看 JCAdCallBackHeader.h

类，请选择所需要的回调状态和参数进行监听和使用

```

[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(msAdLoadCallBack:) name:MSSplashADKey
object:nil];

-(void)msAdLoadCallBack:(NSNotification*)noti{
    NSLog(@"%@",noti.userInfo);
    NSInteger code = [noti.userInfo[@"status"]
integerValue];
    switch (code) {
        case MSAd_splashDidShow:
        {
            NSLog(@"MSAd_splashDidShow");
        }
        break;

        default:
            break;
    }
}

```

3.9、关于欧盟地区展示 GDPR：

欧盟地区需要向用户展示 GDPR 权限获取说明：

```

[JC_iOSAdApi getLocationIsEU:^(BOOL isEU) { //判断是否是欧盟地区
    if (isEU) { //欧盟地区，展示 GDPR 权限界面
        [JC_iOSAdApi jcSDKShowGDPRWithDismissblock:^(

```

```
        } loadFailblock:^(NSError * _Nonnull error) {  
  
        }];  
    }else{  
  
    }  
}];
```

四 、相关报错

如果使用了快手 SDK，在打包上传 AppStore 的时候，苹果不支持模拟器相关支持二进制，可以加入以下脚本，来删除模拟器相关二进制内容。

```
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/$FRAMEWORK_EXEC  
UTABLE_NAME"
```

```
echo "Executable is $FRAMEWORK_EXECUTABLE_PATH"

EXTRACTED_ARCHS=( )

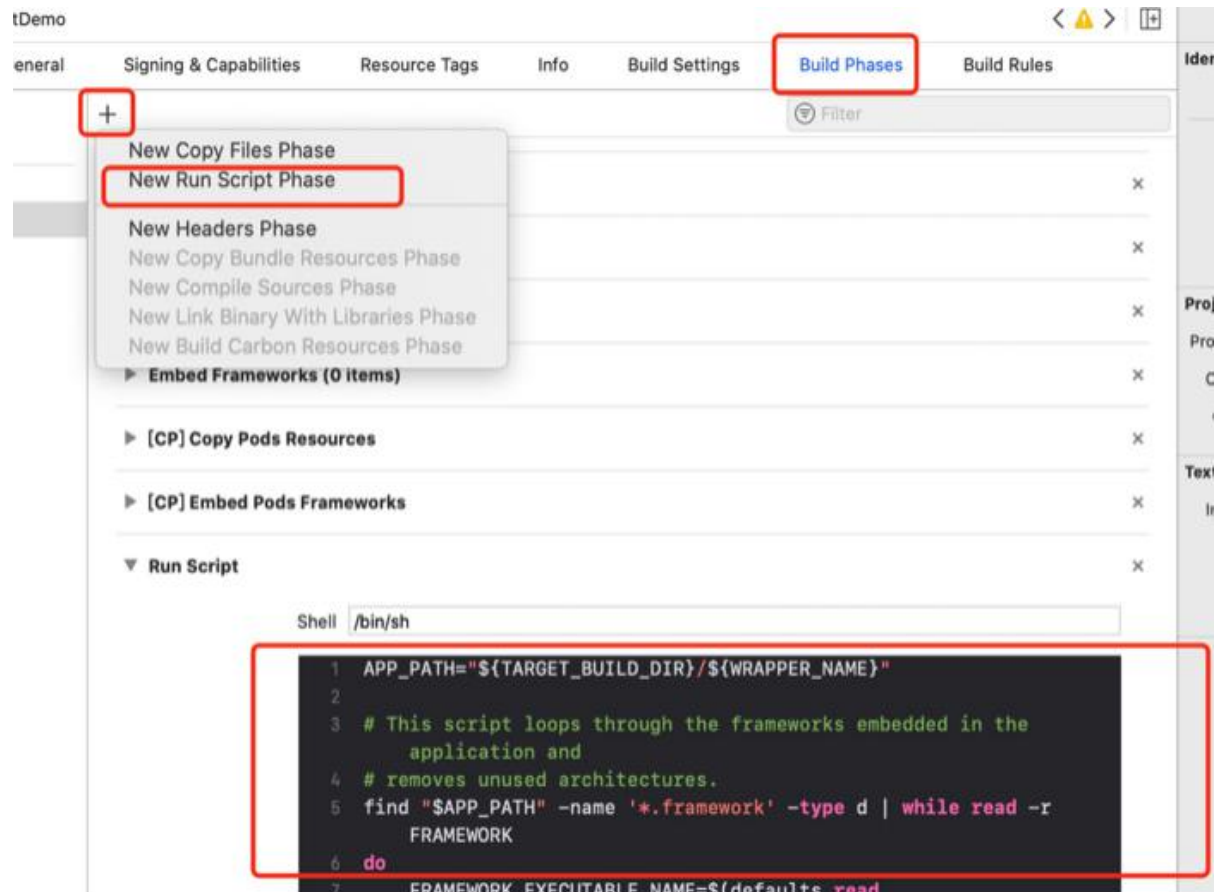
for ARCH in $ARCHS
do
    echo "Extracting $ARCH from
$FRAMEWORK_EXECUTABLE_NAME"
    lipo -extract "$ARCH" "$FRAMEWORK_EXECUTABLE_PATH"
-o "$FRAMEWORK_EXECUTABLE_PATH-$ARCH"
    EXTRACTED_ARCHS+=("$FRAMEWORK_EXECUTABLE_PATH-$ARCH")
done

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。



```
10
11     EXTRACTED_ARCHS=()
12
13     for ARCH in $ARCHS
14     do
15         echo "Extracting $ARCH from $FRAMEWORK_EXECUTABLE_NAME"
16         lipo -extract "$ARCH" "$FRAMEWORK_EXECUTABLE_PATH" -o
17             "$FRAMEWORK_EXECUTABLE_PATH-$ARCH"
18         EXTRACTED_ARCHS+=("$FRAMEWORK_EXECUTABLE_PATH-$ARCH")
19     done
20
21     echo "Merging extracted architectures: ${ARCHS}"
22     lipo -o "$FRAMEWORK_EXECUTABLE_PATH-merged" -create "${EXTRACTED_AR
23     rm "${EXTRACTED_ARCHS[@]}"
```

☒ Show environment variables in build log

☒ Run script only when installing

☐ Use discovered dependency file: `${(DERIVED_FILES_DIR)}/${(INPUT_FILE_PATH)}.d`