About Automation Integration SDK

一、 Project

1、Unity sample project:

If you have downloaded "IOS_SDK_Sample" before reading this document, then please see the steps below

https://github.com/Romambo/JCSDK_Demo

2、Need to download SDK related files separatelyK,

Choose one of the following two download addresses:

Google Cloud Disk:

https://drive.google.com/drive/folders/11N8YLnhyPVxv4HmdGZYni8o59S9aHp7M

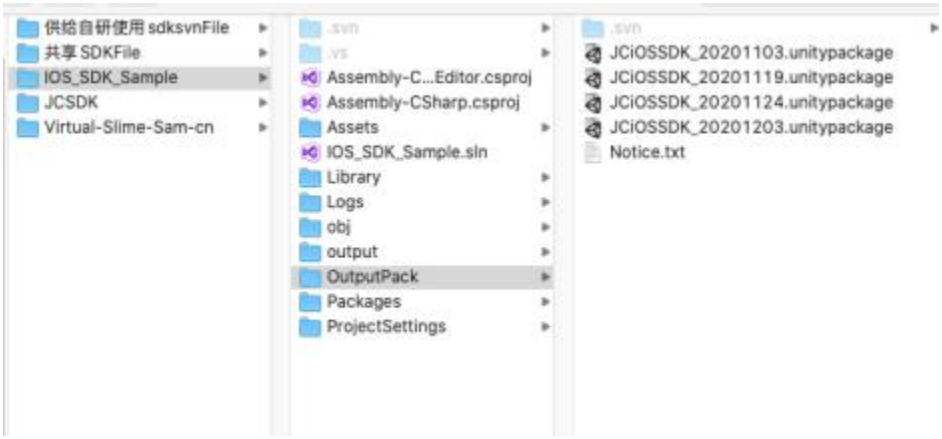
GitHub download path:

https://github.com/Romambo/JCSDK_overseas

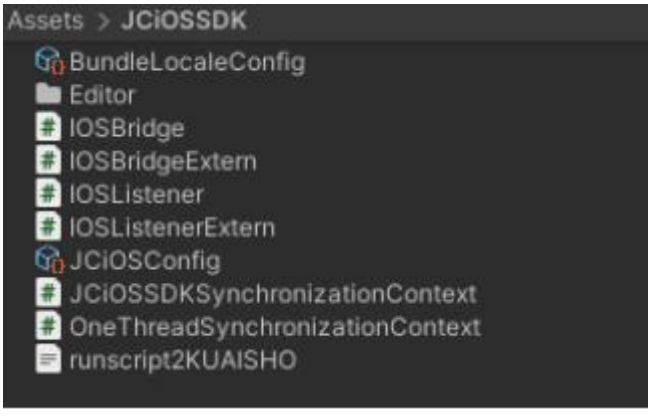
3、To use this integration method。

you only need to import the package in the OutpuPack directory of the sample project in Unity.

Generally, the version with the latest date is sufficient. In some special cases, you can check the description of the corresponding version in Notice.txt to meet the specific requirements, as follows:



4、After importing, all files are in Assets/JCiOSSDK, as follows:

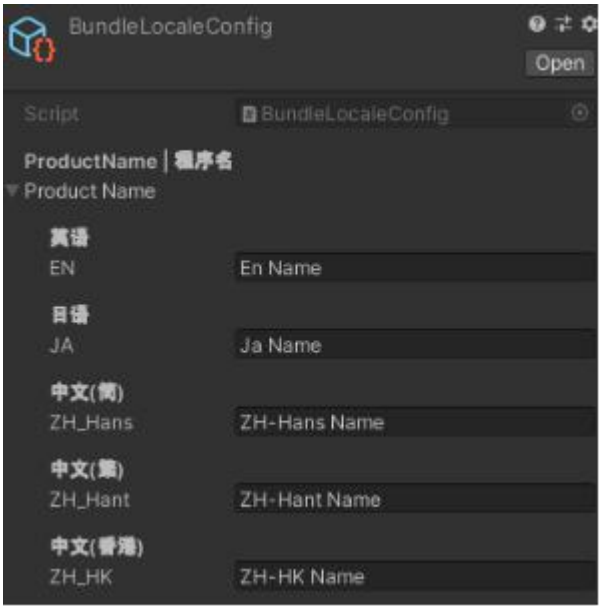


二、 Features

1、Necessary configuration

There are 2 files that need to be configured:

Select the JCiOSSDK/BundleLocaleConfigsset file and set the name displayed in different languages of IOS on the Inspector. There are currently 5 types



Select the JCiOSSDK/JCiOSConfig.asset file and set the relevant SDK parameters on the Inspector。

Special note: here you have to select the SDKFile file you downloaded, first put it in a local and reliable place, and then select it, Please make sure the file directory is correct

[Is USe Old Path] is to be compatible with the export requirements of the old version, which can be ignored

At the same time, it is recommended to set the package name in Unity first to avoid the failure of IOS to automatically add Capability related functions.。

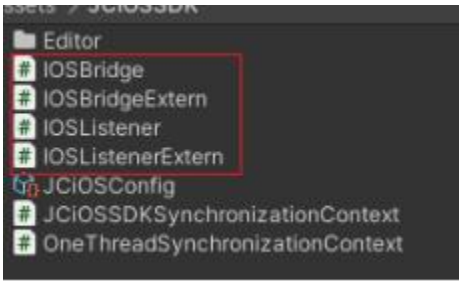
2、 PostProcessBuild

Remarks： The export code probably has done most of the settings of the XCode project, roughly including: adding Capability, adding system frameworks, copying and referencing external frameworks, setting SDK parameters, regular modification of Plist, and modifying the UnityAppController.mm file. And will be accompanied by the SDK upgrade synchronization settings.

See the code for details JCiOSSDKPostprocess.CS

3、 ads Api

For the Api packaging of the advertising SDK, it mainly relies on 4 files:



Among them, IOSBridge is the calling file, and IOSListener is the callback registration file.

3.1、Interface provided by IOSBridge:

bool IsIntersitialReady() ---- Whether interstitial is available

void ShowIntersitial() ---- Show the interstitial (you need to call IsIntersitialReady() first to confirm that the interstitial is available)

bool TryShowIntersitial() ---- Try to show the interstitial, if available, show it, and return to the available state at the same time

bool IsRewardVideoReady()---- Is the reward video available

void ShowRewardVideo()---- show the rewarded video (you need to call IsRewardVideoReady() first to confirm that the rewarded video is available)

bool TryShowRewardVideo()---- Try to show the rewarded video, if available, show it, and return to the available state at the same time

void ShowBanner()---- show Banner

void RemoveBanner()---- Remove Banner

void SendEvent(string eventName) ---- send common dot event

void SendEvent(string eventName, Dictionary<string, string> eventData) ---- send dot event with data

Note: Banner's logic has been encapsulated by SDK, and no longer distinguishes between first display, hiding, and display after destruction.

3.2、Interface provided by IOSListener: (If you are unclear, you can check the remarks on the code)

You can register callback events for related advertising events according to your needs.

At the same time, the multithreading of the callback has been processed, and you can safely operate Unity resources in the registered callback.

It is recommended to encapsulate a layer on top of this project to handle event registration, especially the advertising callback of rewarded video.

The interface is as follows:

```
IOSListener.OnSplashLoadFail += tcb.OnSplashLoadFail;
IOSListener.OnSplashShow += tcb.OnSplashShow;
IOSListener.OnSplashClick += tcb.OnSplashClick;
IOSListener.OnSplashClose += tcb.OnSplashClose;

IOSListener.OnBannerLoaded += tcb.OnBannerLoaded;
IOSListener.OnBannerShow += tcb.OnBannerShow;
IOSListener.OnBannerClick += tcb.OnBannerClick;
IOSListener.OnBannerTapClose += tcb.OnBannerTapClose;
IOSListener.OnBannerAutoRefresh += tcb.OnBannerAutoRefresh;

IOSListener.OnIntersitialLoadFail += tcb.OnIntersitialLoadFail;
IOSListener.OnIntersitialShow += tcb.OnIntersitialShow;
IOSListener.OnIntersitialClick += tcb.OnIntersitialClick;
IOSListener.OnIntersitialClose += tcb.OnIntersitialClose;
IOSListener.OnIntersitialPlayVideoFail += tcb.OnIntersitialPlayVideoFail;
IOSListener.OnIntersitialPlayVideoStart += tcb.OnIntersitialPlayVideoStart;
IOSListener.OnIntersitialPlayVideoEnd += tcb.OnIntersitialPlayVideoEnd;

IOSListener.OnRewardVedioLoadFail += tcb.OnRewardVedioLoadFail;
IOSListener.OnRewardVedioClick += tcb.OnRewardVedioClick;
IOSListener.OnRewardVedioClose += tcb.OnRewardVedioClose;
IOSListener.OnRewardVedioPlayVideoFail += tcb.OnRewardVedioPlayVideoFail;
IOSListener.OnRewardVedioPlayVideoStart += tcb.OnRewardVedioPlayVideoStart;
IOSListener.OnRewardVedioPlayVideoEnd += tcb.OnRewardVedioPlayVideoEnd;
```

三、 Export xcode

After completing the configuration and ensuring that the corresponding interface call and callback processing have been completed in the project code, you can export the XCode project