

# KingSoft Ksyun Mobile Advertising SDK- Unity Plugin For Android Quick Access Document V4.0.3

## ChangeLog

### **Build 4.0.3 [2018/3/13]**

- 1.Renamed the preloadAd interface to loadAd,the same as the name of related callback methods.
- 2.Removed the hasLocalAd interface,now you can use hasAd for all cases.
- 3.Fixed the problem that setSdkEnvironment methods invalid

### **Build 4.0.2 [2018/3/5]**

- 1.Fixed the issue that the close button was occasionally missing
- 2.Fixed the issue that the home key press may occasionally cause the destruction of the video activity
- 3.Modified the package name of FileProvider to avoid name confliction

### **Build 4.0.1 [2018/1/26]**

- 1.Add new interface of hasLocalAd
- 2.Add support for the sandbox environment

### **Build 4.0.0 [2017/12/15]**

- 1.First commit

## Catalog

KingSoft Ksyun Mobile Advertising SDK- Unity Plugin For Android Quick Access  
Document V4.0.3

[ChangeLog](#)

[Catalog](#)

1. List of SDK package contents
2. Integration of SDK

- 3. Support Version
  - 3.1、 Support Version of Unity :
  - 3.2、 Support Version of Android :
- 4. SandBox for testing integration
- 5. Description of the entrance Activity
- 6. Android runtime permissions
- 7. Import SDK
- 8. Quick start guide for the SDK
  - 8.1. Initialization and Ad loading
  - 8.2. Showing Ad
- 9. Advanced usage
  - 9.1. SDK configuration
  - 9.2. Callback of Ad events
  - 9.3. Callback of Ad loading events

## 1. List of SDK package contents

The SDK contains the files as follows:

- Example folder, which including scenes and scripts files related to Demo
- Plugins folder, which containing AndroidManifest file, sdk-aar library file and support-v4-aar library file
- Resources folder, which including image resources related to Demo
- Scripts folder, which including script files of sdk

You can refer to Example/Example.cs, which show the easy way of using sdk.

## 2. Integration of SDK

The SDK is provided as an unitypackage file, including a simple Example.unity.

After importing the SDK project,you must switch the compiling environment of Unity to Android, and then export to the corresponding Android Studio or Eclipse project.

### 3. Support Version

#### 3.1、 Support Version of Unity :

At present, SDK only supports Unity5x or above. If you are using Unity4x version, it is recommended to integrate directly with jar package + asset integration of Android SDK.

#### 3.2、 Support Version of Android :

We can only support minSdkVerion $\geq$ 15(Android 4.0.3) for version of Android.

### 4. SandBox for testing integration

The SDK supports two working environments: the sandbox (SANDBOX\_ENV) and the online (RELEASE\_ENV). The default is the Sandbox. Before initialization, you can change the working environment by means of the SDK configuration items.

It is recommended that customers first use the sandbox environment and the test AppId for developing and testing. After confirm that the interface can work and the data is correct, customers are suggested to switch to the online environment and the corresponding online AppId for the real production.

During the online testing phase, if you frequently encounter an error code (2001) for no advertisement upon requests, the possible reasons are as follows:

1. Low bidding price result in bidding failure
2. Too many requests from a single device exceeds the limit
3. There is no advertising inventory currently

For more detailed support, please contact us

### 5. Description of the entrance Activity

By default, the SDK sets the entry Acitivity to KsyunAdSdkActivity. If you have to implement your own entrance Activity, please comment out the activity tag of KsyunAdSdkActivity in AndroidManifest and add your own entrance Activity tag, then

add the following code to your Activity implementation.

```
1.     protected void onPause () {
2.         KsyunAdSdk.getInstance (). OnPause (this);
3.         super.onPause ();
4.     }
5.
6.     protected void onResume () {
7.         KsyunAdSdk.getInstance (). OnResume (this);
8.         super.onResume ();
9.     }
10.
11.    protected void onDestroy () {
12.        KsyunAdSdk.getInstance (). OnDestroy (this);
13.        super.onDestroy ();
14.    }
```

## 6. Android runtime permissions

By default in Android 6.0 or above system, when initialization, the SDK would request the host app for the following runtime permissions.

- Manifest.permission.READ\_PHONE\_STATE (\*\*Required\*\*for generating a unique ID)
- Manifest.permission.ACCESS\_COARSE\_LOCATION,
- Manifest.permission.ACCESS\_FINE\_LOCATION (\*\*Optional\*\*for geo-location related)

If the APP does not want the SDK to apply for runtime permissions, you can set the corresponding SDK configuration.

## 7. Import SDK

1.Double-Click our sdk unitypackage file, import all of them in the dialog

2.Modify provider tag and file\_paths.xml file

Open the file of Plugins/Android/AndroidManifest, modify the authorities attribute in the provider tag, change the com.xxx.xxx.xxx part to the package name of your app

```
1.     // Add provider here, for compatible of the automatic installation in
2.     // Android 7.0 or above
3.     <provider
4.         android: name = "com.ksc.ad.sdk.util.KsyunFileProvider"
```

```

4.      // attention to the value of the com.xxx part of the authorities below, you need to fill in your own package name
5.      android: authorities = "com.xxx.xxx.xxx.fileprovider"
6.      android: exported = "false"
7.      android: grantUriPermissions = "true">
8.      <meta-data
9.          android: name = "android.support.FILE_PROVIDER_PATHS"
10.         android: resource = "@xml/file_paths" />
11.  </ provider>

```

Modify the Assets/Plugins/Android/res/xml/file\_path.xml file in the SDK root directory, and change the value of path attribute in the external-path tag to Android/data/YOUR\_APP\_PACKAGE\_NAME/

```

1.      // pay attention to the value of the path below, you need to fill in your own package name
2.      <external-path path = "Android/data/com.xxx.xxx.xxx/"
3.          name = "files_root" />
4.      <external-path path = "cache/apk/." name = "external_storage_root" />

```

### 3.Run the project

Depending on your unique needs, there are three possible way to run the project as below.

#### 3.1 Dirrectly build and run the project in Unity Editor Environment

This way is fit for the developer who have no need of modification in the exported android project.

#### 3.2 Export project as Android Studio Module (Supported for the version above Unity3d 5.5.2 )

This way is fit for the developer who already exoprted corresponding android studio project for the intention of modification.

#### 3.3 Export project as Eclipse Android Project (Supported for the version between Unity3d 5.0.0 and 5.5.2 )

This way is fit for the developer who already exported corresponding eclipse android project for the intention of modification.

1. Unity will export the SDK-aar, support-v4-aar Library and Plugins/Android/res folder to the Eclipse Library Project, and export the main Project to Eclipse Android Project.
2. First, you need to import the three library projects mentioned above in Eclipse Workspace, then import the main project, and add dependency on the other three library projects.
3. Then copy the assets folder from the SDK-aar library project into the assets directory of the main project.
4. Modify the class.jar file of library, which automatic generated by unity and may caused dependency conflict. You can refer to the detail message as below.
5. Compile and then run it.

PS: Some detail message of exported Eclipse project by Unity

**If you are using Eclipse project exported by Unity, it will transfer all the aar libraries into the corresponding Eclipse library Projects. You should import and refer all the exported Eclipse library projects. In addition, there are two additional things to do:**

1. All of the jar files generated by the Eclipse Library project are named as class.jar, which may causes dependency conflicts. You should manually rename them.
2. Because the eclipse library project does not support the use of resource files in raw and assets folders, it is necessary to manually copy these folders to the corresponding main project.

## 8. Quick start guide for the SDK

### 8.1. Initialization and Ad loading

If you do not call the method of `setSdkEnvironment ()` to set the SDK environment, the default is the Sandbox.

Please call the loadAd interface in reasonable time after the callback of init success.

```
1.      // Callback for successful initialization
2.      KsyunAdSdk.initSdkSuccess = (string param) => {
3.          Log ("KsyunAdSdk initSdkSuccess");
4.      };
5.      // Callback for failed initialization
6.      KsyunAdSdk.initSdkFailure = (string msg) => {
7.          Log ("KsyunAdSdk initSdkFailure, msg =" + msg);
8.      };
9.      // Initialization method
10.     KsyunAdSdk.init ("YOUR_APP_ID");
11.
12.     //Load the ad in reasonable time
13.     void onLevelStart(){
14.         KsyunAdSdk.loadAd();
15.     }
```

## 8.2. Showing Ad

Before you are going to show your UI item for watching the reward video, we suggest that you first call the hasAd() to check whether the current Ad slot has an advertisement ready for displaying. If the Ad is ready, you can call showAd() to start playing the video for the user.

```
1.      // Check whether there is an advertisement for an Ad slot
2.      if (KsyunAdSdk.hasAd (adSlotId)) {
3.          // Advertisement exists, call showAd interface
4.          KsyunAdSdk.showAd (adSlotId);
5.      } else {
6.          // The advertisement does not exist, you need to call the
loadAd(String adSlotId)
7.          Log ("KsyunAdSdk onNoAd, adSlotId =" + adSlotId);
8.          KsyunAdSdk.loadAd (adSlotId);
9.      }
```

## 9. Advanced usage

### 9.1. SDK configuration

Before calling the method of init(), you can change the environment and some

options by setting the SDK configurations.

```
1.      // Build the SDK configuration class
2.      KsyunAdSdkConfig config = new KsyunAdSdkConfig ();
3.      // Allow the close button to appear during the playback of your re
ward video
4.      config.setShowCloseBtnOfRewardVideo (false);
5.      // Set the waiting time period for showing the close button after
starting the ad video playback
6.      config.setCloseBtnComingTimeOfRewardVideo (5);
7.      // Set SDK to the online environment. the default is the Sandbox
8.      config.setSdkEnvironment (SDK_ENV_SANDBOX);
9.      KsyunAdSdk.init ("YOUR_APP_ID", "YOUR_CHANNEL_ID_IF_NEEDED",
config);
```

## 9.2. Callback of Ad events

By setting callback methods of ad events ,you can monitor the behavior of the user's Ad view.

```
1.      // Callback on successful ad display
2.      public static Action <string> onShowSuccess;
3.      // Callback when ad display failed
4.      public static Action <string> onShowFailed;
5.      // Advertisement content is played, generally for video ads
6.      public static Action <string> onADComplete;
7.      // Ad is clicked
8.      public static Action <string> onADClick;
9.      // The ad is closed
10.     public static Action <string> onADClose;
```

For ads with reward videos, set the setRewardVideoAdListener() interface to check the reward results.

```
1.      // Reward conditions reached
2.      public static Action <string> onADAwardSuccess;
3.      // Reward conditions not reached
4.      public static Action <string> onADAwardFailed;
```

## 9.3. Callback of Ad loading events

By setting Ad loading callback methods, you can monitor the corresponding events as



below.

```
1.      // Ad info loaded successfully, which indicates that we have
      obtained the Ad's information such as name, urls..., but which does no
      t mean that the completion of the download for all video resources
2.      public static Action <string> loadAdInfoSuccess;
3.      // Ad info get failed
4.      public static Action <string> loadAdInfoFailure;
5.      // Downloading for resources completed, the parameter is the adver
      tising Id
6.      public static Action <string> onAdLoaded;
```