

# KingSoft Ksyun Mobile Advertising SDK- Unity Plugin For Android Quick Access Document V4.0.2

## ChangeLog

### **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 V4.0.2

[ChangeLog](#)

[Catalog](#)

- [1. List of SDK package contents](#)
- [2. Integration of SDK](#)
- [3. About Eclipse project exported by Unity](#)
- [4. Support Version](#)
- [5. SandBox for testing integration](#)
- [6. Description of the entrance Activity](#)
- [7. Android runtime permissions](#)
- [8. Import SDK](#)

- 9. Quick start guide for the SDK
  - 9.1. Initialization and Preloading
  - 9.2. Showing Ad
- 10. Advanced usage
  - 10.1. SDK configuration
  - 10.2. Callback of Ad events
  - 10.3. Callback of preloading events
  - 10.4. About preload

## 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. About Eclipse project exported 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 rojects. 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.

## 4. Support Version

**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**

## 5. 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

## 6. 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.    }
```

## 7. 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.

## 8. Import SDK

1. Double-Click our sdk unitypackage file, import all of them in the dialog
2. 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.      <provider
3.          android: name = "com.ksc.ad.sdk.util.KsyunFileProvider"
4.          // attention to the value of the com.xxx part of the authorities b
        elow, 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>

```

### 3.Export the corresponding Android project

Depending on the type of Android project exported, there are two possible integration ways as below.

#### 3.1 Export project as Android Studio Module.

Modify the 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/, and place it in the res folder under the exported Android Studio project directory, and then run it.

```

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"
        />

```

#### 3.2 Export project as Eclipse Android Project.

1. Unity will export the SDK -aar and support-v4-aar Library to the Eclipse Library Project, and export the main Project to Eclipse Android Project.
2. First, you need to import the three projects mentioned above in Eclipse Workspace, and add the main project dependency on the other two library projects.
3. Then copy the assets folder from the SDK -aar library project into the assets directory of the main project.
- 4, last modified the main project's res/XML/file\_paths , and then change the value of

path attribute in the external-path tag to  
Android/data/YOUR\_APP\_PACKAGE\_NAME/ , and then run it.

```
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"
      />
```

## 9. Quick start guide for the SDK

### 9.1. Initialization and Preloading

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

```
1.      // Callback for successful initialization
2.      KsyunAdSdk.initSdkSuccess = (string param) => {
3.          Log ("KsyunAdSdk initSdkSuccess");
4.          // pre-load ad contents for all adslots
5.          KsyunAdSdk.preloadAd ();
6.      };
7.      // Callback for failed initialization
8.      KsyunAdSdk.initSdkFailure = (string msg) => {
9.          Log ("KsyunAdSdk initSdkFailure, msg =" + msg);
10.     };
11.     // Initialization method
12.     KsyunAdSdk.init ("YOUR_APP_ID");
```

### 9.2. Showing Ad

Before you are going to show your UI item for watching the reward video, we suggest that you first call the `hasLocalAd()` 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.hasLocalAd (adSlotId)) {
3.          // Advertisement exists, call showAd interface
```

```

4.         KsyunAdSdk.showAd (adSlotId);
5.     } else {
6.         // The advertisement does not exist, you need to call the
        preloadAd(String adSlotId) for preloading
7.         Log ("KsyunAdSdk onNoAd, adSlotId =" + adSlotId);
8.         KsyunAdSdk.preloadAd (adSlotId);
9.     }

```

## 10. Advanced usage

### 10.1. 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);

```

### 10.2. 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;
```

### 10.3. Callback of preloading events

By setting preload callback methods, you can monitor corresponding events for Ad preloading.

```
1.      // Ad info preloaded successfully, which indicates that we have
      // obtained the Ad's information such as name, urls..., but which does not
      // mean that the completion of the download for all video resources
2.      public static Action <string> preloadAdInfoSuccess;
3.      // Ad info get failed
4.      public static Action <string> preloadAdInfoFailure;
5.      // Downloading for resources completed, the parameter is the advertising Id
6.      public static Action <string> onAdLoaded;
```

### 10.4. About preload

- If you want to play only local cached videos, use the `hasLocalAd` method before calling the `ShowAd` method. This method returns true only if there is a cached video for the corresponding ad slot.
- If the `hasLocalAd` method returns false, the App is required to call the `preloadAd` method for ad pre-loading targeting on the specific ad slot.