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

ChangeLog

Build 4.0.3 [2018/3/9]
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.

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

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 Acitivty 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.

```
    protected void onPause () {
        KsyunAdSdk.getInstance (). OnPause (this);
        super.onPause ();
    }

    protected void onResume () {
        KsyunAdSdk.getInstance (). OnResume (this);
        super.onResume ();
    }

    protected void onDestroy () {
        KsyunAdSdk.getInstance (). OnDestroy (this);
        super.onDestroy ();
    }
```

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

```
// Add provider here, for compatible of the automatic installation in Android 7.0 or above
<provider
    android: name = "com.ksc.ad.sdk.util.KsyunFileProvider"
    // attention to the value of the com.xxx part of the authorities below, you need to fill in your own package name
    android: authorities = "com.xxx.xxx.xxx.fileprovider"
    android: exported = "false"
    android: grantUriPermissions = "true">
    <meta-data
        android: name = "android.support.FILE_PROVIDER_PATHS"
        android: resource = "@xml/file_paths" />
</ 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.

```
    // pay attention to the value of the path below, you need to fill in your own package name
    <external-path path = "Android/data/com.xxx.xxx.xxx/"
            name = "files_root" />
    <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.

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

9. Quick start guide for the SDK

9.1. 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.

```
    // Callback for successful initialization
    KsyunAdSdk.initSdkSuccess = (string param) => {
        Log ("KsyunAdSdk initSdkSuccess");
    };
    // Callback for failed initialization
    KsyunAdSdk.initSdkFailure = (string msg) => {
        Log ("KsyunAdSdk initSdkFailure, msg =" + msg);
    };
    // Initialization method
    KsyunAdSdk.init ("YOUR_APP_ID");

    //Load the ad in reasonable time
    void onLevelStart(){
        KsyunAdSdk.loadAd();
    }
```

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 haslAd() 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.

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

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.

```
    // Build the SDK configuration class
    KsyunAdSdkConfig config = new KsyunAdSdkConfig ();
    // Allow the close button to appear during the playback of your re
ward video
    config.setShowCloseBtnOfRewardVideo (false);
    // Set the waiting time period for showing the close button after
starting the ad video playback
    config.setCloseBtnComingTimeOfRewardVideo (5);
    // Set SDK to the online environment. the default is the Sandbox
    config.setSdkEnvironment (SDK_ENV_SANDBOX);
    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.

```
    // Callback on successful ad display
    public static Action <string> onShowSuccess;
    // Callback when ad display failed
    public static Action <string> onShowFailed;
    // Advertisement content is played, generally for video ads
    public static Action <string> onADComplete;
    // Ad is clicked
    public static Action <string> onADClick;
    // The ad is closed
    public static Action <string> onADClose;
```

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

```
    // Reward conditions reached
    public static Action <string> onADAwardSuccess;
    // Reward conditions not reached
    public static Action <string> onADAwardFailed;
```

10.3. Callback of Ad loading events

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

```
    // Ad info loaded 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
    public static Action <string> loadAdInfoSuccess;
    // Ad info get failed
    public static Action <string> loadAdInfoFailure;
    // Downloading for resources completed, the parameter is the advertising Id
    public static Action <string> onAdLoaded;
```