SDK Unity Plugin - An Access Guideline by Pangle Ad Network

Document Version	Revision Date	Revision Statement
v3.3.0.2	2020-11-30	RewardVideo AD SetMediaExtra invalid problem fixed
v3.3.0.1	2020-11-20	Fixed a problem with the successful proxy invocation of the video cache for iOS side
v3.3.0.0	2020-11-05	Support Unity2020. Add support for native ads. Fix known bugs.
v2.9.0.7	2020-05-09	iOS adapts to Unity2019.3 + (see 1.5.2 and 1.5.3 for changes)
v2.7.5.2	2020-01-09	Fixed known bugs
v2.7.5.1	2019-12-27	Added support for express template ads (ExpressFeed, ExpressInterstitial, and ExpressBanner)
v2.3.0.8	2019-09-25	Improved compatibility with Unity and strengthened security
v2.3.0.7	2019-09-23	Added express ads
v1.9.9.2	2019-09-25	Improved callback of full-screen video ads, enhanced underlying logic protection rewarded- video ads and full-screen video ads
v1.9.9.1	2019-02-27	Fixed rare bugs during download, and other optimizations
v1.0.0.0	2020-4-16	Basic mediation functions
v1.0.0.0	2020-4-16	Basic mediation functions
v1.0.0.0	2020-4-16	Basic mediation functions
v1.9.9.0	2019-01-10	Scene optimization for videos, style optimization for rewarded video ads

1.Add SDK Unity to Pangle Ad Network

This guide is for developers who want to profit from their Unity apps.

To display ads from Pangle Ad Network and earn revenue, the first step is to integrate the Unity plugin for Pangle Ad Network mobile ads into the app. Once the integration is complete, you can choose an ad style to access, such as a Native Ad, a rewarded video ad, or a full-screen video ad. Here are the detailed access steps.

1.1 Import of Unity plugin

1.1.1 Apply for App CodeId

Please apply for the test app ID and ad slot ID from relevant Pangle Ad Network personnel.

1.1.2 Import Pangle Ad Network Unity Plugin

With the Pangle Ad Network Unity plugin, Unity developers can easily display ads on Android and iOS apps without writing Java or Objective-C code.
Open your project in the Unity editor and select Assets > Import Package > Custom Package and find the PangleAdapterScripts.unitypackage and PangleSDK.unitypackage that you downloaded. Make sure to select all files and click Import to import.

Example.unitypackage - Advertising scene access example

PangleAdapterScripts.unitypackage - Contains Unity plugin file used to use the SDK in Unity

PangleSDK.unitypackage - Contains Pangle‘s AD SDK

1.2 Android Network Configuration

After importing the Pangle Ad Network Unity plugin, check the following configuration information in the corresponding AndroidManifest.xml file on the Android platform. After configuring AndroidManifest.xml, you need to check the configuration of the obfuscated file.

1.2.1 Add Permissions

```
    <!-- Required Permissions -->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <!-- If there is a video ad and it is played with textureView, please be
sure to add this, otherwise a black screen will appear -->
    <uses-permission android:name="android.permission.WAKE_LOCK" />
```

1.2.2 Adapt to Android 7.0 and Above

If your app needs to adapt to Android 7.0 and above, add the following code to AndroidManifest

```
<provider
     android:name="com.bytedance.sdk.openadsdk.TTFileProvider"
     android:authorities="${applicationId}.TTFileProvider"
     android:exported="false"
     android:grantUriPermissions="true">
    <meta-data
         android:name="android.support.FILE_PROVIDER_PATHS"
         android:resource="@xml/file_paths" />
</provider>
```

In the Assets/Plugins/Android/res/xml directory, create a new xml file file_paths and add the following code to the file:

```
<?xml version="1.0" encoding="utf-8"?>
<paths xmlns:android="http://schemas.android.com/apk/res/android">
     <external-files-path name="external_files_path" path="Download" />
     <!--In order to adapt all paths you can set path = "." -->
</paths>
```

```
 In order to adapt the download and installations,when using package
com.android.support: support-v4:24.2.0 in the project,
 Please use 24.2.0 and above.
```

Note the sdk that the plugin depends on:

1：Support related packages：
In Assets/UnionPlatform/Plugins/Android, the plugin directory provided by us, there are support-v4-compat.jar and support-v4-utils.jar. If the developer also uses similar packages, you can choose your own support package, note the version of the package needs to be 24.2.0 and above.

2：Ad plugin related packages：
In Assets/UnionPlatform/Plugins/Android, the plugin directory we provide, there is open_ad_sdk.aar which must be introduced.

3：The package used in Native Ad customization:
Assets/UnionPlatform/Plugins/Android, the plugin directory provided by us, volley.jar is useful for loading custom images in Native Ads. Developers can replace it with their own library according to the images they use, and do the corresponding modification in the native implementation category.

1.2.3 Provider Configuration

Note: It is required for both single-process and multi-process environment.

```
<provider
    android:name="com.bytedance.sdk.openadsdk.multipro.TTMultiProvider"
    android:authorities="${applicationId}.TTMultiProvider"
    android:exported="false" />
```

1.2.4 Operating Environment Configuration

This plugin runs on Android 4.0 (API Level 14) and above on the Android platform.

```
<uses-sdk android:minSdkVersion="14" android:targetSdkVersion="24" />
```

1.2.5 Precautions for Android

1. New OnRefuse interface on IDislikeInteractionListener to prevent mutil dislike dialog popup
2. please use TextureView instead of SurfaceView for support native video ads. Code Sample:

```

```

3. Create a new script 'ModifyUnityAndroidAppManifestSample' under folder 'Assets/Editor'

1.2.6 Precautions for iOS

1. PangleTools file added. PangleTools allows you to get the screen height, width, and safeAreaInsets up, down, left and right of window .

Note:

1:If the developer declares target SdkVersion to be API 23 and above, be sure to apply for all permissions required by the SDK before calling any of the interfaces of this SDK. Otherwise the SDK may not work properly.

2:If targetSdkVersion >= P version, you need to add the following in the AndroidManifest.xml file:

```
<uses-libraryandroid:name="org.apache.http.legacy"android:required="false"/>
```

Note: Five types of SO files are supported in this SDK:x86,x86_64,armeabi,armeabi-v7a,arm64-v8a , If your app does not support one of these, please use abiFilters in build.gradle to select a supported architecture as follows:
ndk {
// Set the supported SO library. Please note to set abiFilters 'armeabi-v7a', 'arm64-v8a', 'x86', 'x86_64','armeabi' accordingly.
}

1.3 Obfuscation

When generating the Android platform release installation package, you need to check the obfuscated configuration. If you need to use Proguard to obfuscate your code, be sure not to obfuscate the SDK code. Please add the following configuration at the end of the proguard-user.txt file (or other obfuscated files):

```
#Obfuscation required for ad sdk
-keep class com.bytedance.sdk.openadsdk.** { *; }
-keep public interface com.bytedance.sdk.openadsdk.downloadnew.** {*;}
-keep class com.pgl.sys.ces.* {*;}

#The following class is a custom java class for Native Ads. It needs to be
kept. When the developer implements it, it also needs to be kept.
-keep class com.bytedance.android.NativeAdManager {*;}
-keep class com.bytedance.android.IntersititialView {*;}
-keep class com.bytedance.android.BannerView {*;}
```

Note: If the SDK code is obfuscated, it will cause the ad to fail to display or other anomalies.

For Unity's Player settings on the Android platform, please refer to: Android Player settings

1.4 Initialize Unity Plugin on Android

Initialization on the Android platform

The developer needs to call the following code in the Application#onCreate() method to initialize the Pangle Ad Network SDK. The current SDK supports multiple processes. If it is a multi-process environment, you need to turn on the relevant settings according to the instructions.

```
public class UnionApplication extends Application {

    @Override
    public void onCreate() {
        super.onCreate();

        //It is strongly recommended to call the corresponding Application
#onCreate () method, to avoid content shown as null
        TTAdSdk.init(context,
                            new TTAdConfig.Builder()
                                    .appId("5001121")
                                    .useTextureView(false) //Use TextureView
to play the video. The default setting is SurfaceView, when the
                                    context is in conflict with SurfaceView,
you can use TextureView
                                    .appName("APP Test Media")

.titleBarTheme(TTAdConstant.TITLE_BAR_THEME_DARK)
                                    .allowShowNotify(true) //Allow or deny
permission for SDK to display notification bar
                                    .allowShowPageWhenScreenLock(true) //Allow
or deny permission to display the landing page ad in the lock
                                      screen
```

```
                                    .debug(true) //Turn it on during the
testing phase, you can troubleshoot with the log, remove it after launching
the
                                      app

.directDownloadNetworkType(TTAdConstant.NETWORK_STATE_WIFI,
TTAdConstant.NETWORK_STATE_3G) //Collections of network states that allow
direct download
                                    .supportMultiProcess(false) //Whether to
support multi-process, true indicates support

.customController(getController())//Control private data
                                    .build());
     }

     private static TTCustomController getController() {
        MyTTCustomController customController = new MyTTCustomController();
        return customController;
    }

    private static class MyTTCustomController extends TTCustomController{
        @Override
        public boolean isCanUseLocation() {
            //Allow or deny SDK to actively use geographic location
information
            return super.isCanUseLocation();
        }

        @Override
        public TTLocation getTTLocation() {Yes/No
            //When isCanUseLocation = false, user's geographic location
information will be passed in. Pangle SDK will use the geographic location
information that you passed in.
            return super.getTTLocation();
        }

        @Override
        public boolean alist() {
            return super.alist();
        }

        @Override
        public boolean isCanUsePhoneState() {
            //Allow or deny SDK to use user's mobile phone hardware
parameters, such as IMEI
            return super.isCanUsePhoneState();
        }

        @Override
```

```
        public String getDevImei() {
            //When isCanUsePhoneState = false, user's IMEI information will be
passed in. Pangle SDK will use the IMEI information that you passed in.
            return super.getDevImei();
        }

        @Override
        public boolean isCanUseWifiState() {
            //Allow or deny SDK to use the ACCESS_WIFI_STATE permission
            return super.isCanUseWifiState();
        }

        @Override
        public boolean isCanUseWriteExternal() {
            //Allow or deny SDK to use the WRITE_EXTERNAL_STORAGE permission
            return super.isCanUseWriteExternal();
        }
    }
}
```

1.4.1 Initialization Interface

```
    /**
     * Entrance of Pangle Ad Network SDK initialization
     *
     * @param context Mush be application context
     * @param config  Initialize configuration and required parameters
     * @return TTAdManager example
     */
    public static TTAdManager init(Context context, TTAdConfig config);
```

1.4.2 Initialization Configuration Parameter:

```
    public static class TTAdConfig.Builder {
        private String mAppId;// Required parameter, set the app's AppId
        private String mAppName;// Required parameter, set the app name
        private boolean mIsPaid = false;// Optional parameter, set whether it
is a paid user: true indicates paid user, false indicates non-paying user.
Default setting is false, non-paying user
        private int mGender = TTAdConstant.GENDER_UNKNOWN;//Optional
parameter, set the user gender. The default setting is unknown
TTAdConstant#GENDER_UNKNOWN
        private int mAge;// Optional parameter, set user age ** must be
greater than 0**
        private String mKeywords;// Optional parameter, set the keyword list
for user profile** can't exceed 1000 characters**
        private String mData;// Optional parameter, set additional user
information** can't exceed 1000 characters**
```

```
        private int mTitleBarTheme =
TTAdConstant.TITLE_BAR_THEME_LIGHT;//Optional parameter, set the landing page
theme, the default setting is TTAdConstant#TITLE_BAR_THEME_LIGHT
        private boolean mAllowShowNotify = true;//Optional parameter, set
whether to allow SDK popup notification: true indicates allow, false indicates
do not allow. The default setting is true, allow
        private boolean mIsDebug = false;//Optional parameter, whether to
enable debug information output: true indicates enable, false indicates
disable. The default setting is false, disable
        private boolean mAllowShowPageWhenScreenLock = false;//Optional
parameter, set whether to allow landing pages to appear on the lock screen:
true indicates allow, false indicates do not allow. The default setting is
false
        private int[] mDirectDownloadNetworkType;
        private boolean mIsUseTextureView = false;//Optional parameter, set
whether to use texture to play the video: true indicates use, false indicates
do not use. The default setting is false, do not use (the default setting is
to use surface)
        private boolean mIsSupportMultiProcess = false;//Optional parameter,
set whether to support multi-process: true indicates support, false indicates
do not support. The default setting is false, do not support
        private TTCustomController mCustomController;//Optional parameter, you
can set the privacy information control switch

    }
```

1.5 nitialize Unity Plugin on iOS

1.5.1 Configuring the startup function `UnionAppController.mm`, initializing SDK must be done before loading the ad

```
[BUAdSDKManager setAppID:@"5000546"];
[BUAdSDKManager setIsPaidApp:NO];
```

1.5.2 Third-party library configuration that needs to be relied upon is in the `XCodePostProcess` file, you can import it directly. If

you have the file locally, you need to add the following code to all configurations.

```
var projPath = pathToBuiltProject + "/Unity-iPhone.xcodeproj/project.pbxproj";
                var proj = new PBXProject();
                proj.ReadFromFile(projPath);
            var targetGUID = @"";
#if (UNITY_2019_3_OR_NEWER)
            // New API for 2019.3 or newer version
```

```
            targetGUID = proj.GetUnityFrameworkTargetGuid();
#else
            // Previous API
            targetGUID = proj.TargetGuidByName("Unity-iPhone");
#endif
            proj.AddBuildProperty(targetGUID, "OTHER_LDFLAGS", "-ObjC");
                proj.AddFrameworkToProject(targetGUID, "libresolv.9.tbd",
false);
                proj.AddFrameworkToProject(targetGUID, "libc++.tbd", false);
                proj.AddFrameworkToProject(targetGUID, "libz.tbd", false);
                proj.AddFrameworkToProject(targetGUID,
"CoreLocation.framework", false);
                proj.AddFrameworkToProject(targetGUID, "AVKit.framework",
false);
                proj.AddFrameworkToProject(targetGUID, "ImageIO.framework",
false);
                proj.AddFrameworkToProject(targetGUID,
"MediaPlayer.framework", false);
                proj.AddFrameworkToProject(targetGUID, "StoreKit.framework",
false);
                proj.AddFrameworkToProject(targetGUID,
"SystemConfiguration.framework", false);
                proj.AddFrameworkToProject(targetGUID, "AdSupport.framework",
false);
                proj.AddFrameworkToProject(targetGUID, "CoreMotion.framework",
false);
                proj.AddFrameworkToProject(targetGUID, "Photos.framework",
false);
                proj.AddFrameworkToProject(targetGUID, "WebKit.framework",
false);
                proj.AddFrameworkToProject(targetGUID,
"CoreTelephony.framework", false);
                proj.AddFrameworkToProject(targetGUID,
"CoreServices.framework", false);
                proj.AddFrameworkToProject(targetGUID, "Security.framework",
false);
                proj.AddFrameworkToProject(targetGUID, "Accelerate.framework",
false);
                proj.AddFrameworkToProject(targetGUID, "libsqlite3.tbd",
false);
                proj.AddFrameworkToProject(targetGUID, "libbz2.tbd", false);
                proj.AddFrameworkToProject(targetGUID, "libxml2.tbd", false);
                proj.WriteToFile(projPath);
```

1.5.3 Bundle Configuration Related Instructions

- 1.If the SDK and bundle files are first placed in a Unity project (such as this Demo) without configuration errors, then export the Xcode project. At this time, the bundle file will be

automatically imported into Build Phases-Copy Bundle Resources.

- 2.When the Unity version is greater than 2019.3, you need to make sure that the bundle file is added to the main target of the Xcode project, not the UnityFramework, and select Copy if needed when adding.
- 3.To ensure the normality of the bundle file, be sure to check that the bundle file has been configured to Build Phases-Copy Bundle Resources after export.
- 4.If you use Pod to introduce SDK after exporting project, make sure that the copy script of the Bundle file in the Pod directory (usually added automatically) under the main Target-Copy Pods Resources is normal. This configuration cannot coexist with 3. If you choose 3, you need to copy the Bundle to the main Target project directory (Copy if needed).

2.Load Ad

2.1 Build AdNative Object:

The AdNative object is the interface to access the entire SDK and can be used for ad acquisition.

Construction method:

```
//Must be called after initialization, otherwise it will be null
 this.adNative = SDK.CreateAdNative();
```

SDK Interface:

```
 public static class SDK
    {
        /// <summary>
        /// Get the version number of the SDK
        /// </summary>
        public static string Version
        {
            get { return "1.0.0"; }
        }

        /// <summary>
        /// Create an AdNative object to load the ad
        /// </summary>
        public static AdNative CreateAdNative()
        {
            return new AdNative();
        }

        /// <summary>
        /// On the Android platform, request some necessary permissions, such
as READ_PHONE_STATE.
        ///To get the best ads for you, and to increase the fill rate for
rewarded video ads and app install ads, etc.
```

```
        ///It is recommended to call this method at the appropriate time
before the ad request.
        ///Call the following method when the user first launches the main
interface of your app:
        /// </summary>
        public static void RequestPermissionIfNecessary()
        {
        }

        /// <summary>
        /// Whether to show the install application dialog when exiting the
app
        /// </summary>
        /// <returns>True stands for to show the dialog.</returns>
        public static bool TryShowInstallDialogWhenExit(Action onExitInstall)
        {
            return false;
        }
    }
```

The AdNative Interface:

```
public sealed class AdNative
    {
        /// <summary>
        /// Load the Native Ad.
        /// </summary>
        public void LoadFeedAd(AdSlot adSlot, IFeedAdListener listener)
        {
        }

        /// <summary>
        /// Load Vertical In-feed Video Ad
        /// </summary>
        public void LoadDrawFeedAd(AdSlot adSlot, IDrawFeedAdListener
listener)
        {
        }

        /// <summary>
        /// Load Native Ads, currently supports native banner, native banner.
        /// </summary>
        public void LoadNativeAd(AdSlot adSlot, INativeAdListener listener)
        {
        }

        /// <summary>
```

```
        /// Load the template banner ad.
        /// </summary>
        public void LoadBannerAd(AdSlot adSlot, IBannerAdListener listener)
        {
        }

        /// <summary>
        /// Load template interstitial ads
        /// </summary>
        public void LoadInteractionAd(
            AdSlot adSlot, IInteractionAdListener listener)
        {
        }

        /// <summary>
        /// Load splash ad, set the timeout
        /// </summary>
        public void LoadSplashAd(
            AdSlot adSlot, ISplashAdListener listener, int timeOut)
        {
        }

        /// <summary>
        /// Load splash ad
        /// </summary>
        public void LoadSplashAd(AdSlot adSlot, ISplashAdListener listener)
        {
        }

        /// <summary>
        /// Load rewarded video ad
        /// </summary>
        public void LoadRewardVideoAd(
            AdSlot adSlot, IRewardVideoAdListener listener)
        {
        }

        /// <summary>
        /// Load full-screen video ad
        /// </summary>
        public void LoadFullScreenVideoAd(
            AdSlot adSlot, IFullScreenVideoAdListener listener)
        {
        }

        /// <summary>
        /// Load express rewarded video ad
        /// </summary>
        public void LoadExpressRewardAd(
```

```
            AdSlot adSlot, IRewardVideoAdListener listener)
        {
        }

        /// <summary>
        /// Load express full-screen video ad
        /// </summary>
        public void LoadExpressFullScreenVideoAd(
           AdSlot adSlot, IFullScreenVideoAdListener listener)
        {
        }

        /// <summary>
        /// Load express native ad
        /// </summary>
        public void LoadNativeExpressAd(
            AdSlot adSlot, IExpressAdListener listener)
        {
        }

        /// <summary>
        /// Load express interstitial ad
        /// </summary>
        public void LoadExpressInterstitialAd(
            AdSlot adSlot, IExpressAdListener listener)
        {
        }
```

```
    ///
/// Load express banner ad  ///
public void LoadExpressBannerAd(  AdSlot adSlot, IExpressAdListener listener) { } }
```

2.2 Build AdSlot Object

The AdSlot object is the ad information that needs to be set when ADNative loads the ad.

Construction method:

```
AdSlot adSlot = new AdSlot.Builder()
    // Required parameter, set your CodeId
    .setCodeId("900486272")
    // Required parameter, set the maximum size of the ad image and the
    desired aspect ratio of the image, in units of px
    //Note: If you select a native ad on the Pangle Ad Network, the returned
image size may differ significantly from the size you expect
    .setImageAcceptedSize(640, 320)
    // required parameter for Express native ad, express interstitial ad, and
express banner ad. In units of dp.
```

```
    .SetExpressViewAcceptedSize(400, 80)
    // Optional parameter, allow or deny permission to support deeplink
    .setSupportDeepLink(true)
    // Optional parameter, set the number of ads returned per request for in-
feed ads, up to 3
    .setAdCount(2)
    //Required parameter for native ad requests, choose TYPE_BANNER or
    TYPE_INTERACTION_AD
    .setNativeAdType(AdSlot.TYPE_BANNER)
    //Parameter for rewarded video ad requests, name of the reward
    .setRewardName("gold coin")
    //The number of rewards in rewarded video ad
    .setRewardAmount(3)
    //User ID, a required parameter for rewarded video ads
    //It is developer's unique identifier for users; If sdk pass-through is
not necessary or the server is not in callback mode, it can be set to an empty
string
    .setUserID("user123")
    //Set how you wish the video ad to be displayed, choose from
TTAdConstant.HORIZONTAL or TTAdConstant.VERTICAL
    .setOrientation(orientation)
    //Pass-through parameters and strings of rewards in rewarded video ad, if
you use json object, you must use serialization as String type, it can be
empty
    .setMediaExtra("media_extra")
    .build();
```

2.4 Access Native Ad

The SDK provides native ad for the access party. Currently, it supports self-rendering banner ads and interstitial ads. Since the advertising scenario requires the developer to render the ad view and needs to be customized based on the Android or iOS platform, the demo we provide is a related usage example.

2.4.1 Load Native Ad

The access party can call AdNative's `LoadNativeAd(AdSlot adSlot, INativeAdListener listener)` to load native ads, adslot is the required information for requesting ads, and listener is the callback of a successful or failed ad loading.

Note: For SetNativeAdType method of the adSlot parameter, AdSlotType.Banner or AdSlotType.InteractionAd must be passed.

The following example loads a native banner ad:

```
    public void LoadNativeNannerAd()
    {
```

```
#if UNITY_IOS
        if (this.bannerAd != null)
        {
            Debug.LogError("Ads are already loaded");
            this.information.text = "Ads are already loaded";
            return;
        }
#else
        if (this.mBannerAd != null)
        {
            Debug.LogError("Ads are already loaded");
            this.information.text = "Ads are already loaded";
            return;
        }
#endif

        var adSlot = new AdSlot.Builder()
#if UNITY_IOS
            .SetCodeId("900546687")
#else
            .SetCodeId("901121423")
#endif
            .SetSupportDeepLink(true)
            .SetImageAcceptedSize(600, 257)
            .SetNativeAdType(AdSlotType.Banner)
            .SetAdCount(1)
            .Build();
        this.AdNative.LoadNativeAd(adSlot, new NativeAdListener(this));
    }
```

The implementation of NativeAdListener is as follows:

```
 private sealed class NativeAdListener : INativeAdListener
    {
        private Example example;

        public NativeAdListener(Example example)
        {
            this.example = example;
        }

        public void OnError(int code, string message)
        {
            Debug.LogError("OnNativeAdError: " + message);
            this.example.information.text = "OnNativeAdError: " + message;
        }

         //In the callback method, list shows the ads that are loaded for the
Android platform, and ad shows the ads loaded for the iOS platform.
```

```
         //Note that the two platforms use different objects.
        public void OnNativeAdLoad(AndroidJavaObject list,NativeAd ad)
        {
#if UNITY_IOS
            //iOS platform to get native ad objects
            this.example.bannerAd = ad;
            this.example.intersititialAd = ad;
            ad.SetNativeAdInteractionListener(
                new NativeAdInteractionListener(this.example)
            );
#else
            var size = list.Call<int>("size");

           //Android platform to get native ad objects
            if(size > 0)
            {
                this.example.mBannerAd = list.Call<AndroidJavaObject>("get",
0);
                this.example.mIntersititialAd = list.Call<AndroidJavaObject>
("get", 0);
            }

#endif
            Debug.Log("OnNativeAdLoad");
            this.example.information.text = "OnNativeAdLoad";
        }
    }
```

2.4.2 Display Native Ad

After loading the native ad, you need to rely on the platform to customize the native ad view; the iOS platform calls the native ad's ShowNativeAd method to display the ad; the Android platform calls the custom implementation display method. (In the example, a native banner ad is used as an example)

```
  public void ShowNativeNannerAd()
    {
#if UNITY_IOS
       if (bannerAd == null)
        {
            Debug.LogError("Please load the ad first");
            this.information.text = "Please load the ad first";
            return;
        }
        this.bannerAd.ShowNativeAd();
#else
        if (mBannerAd == null)
```

```
        {
            Debug.LogError("Please load the ad first");
            this.information.text = "Please load the ad first";
            return;
        }
        if (mNativeAdManager == null)
        {
            mNativeAdManager = GetNativeAdManager();
        }
        mNativeAdManager.Call("showNativeBannerAd", activity, mBannerAd);
#endif
    }
```

The mNativeAdManager is obtained by obtaining an instance object of the java class com.bytedance.android.NativeAdManager on the Android platform, and the developer can customize the implementation method, and the obtaining method is as follows:

```
 public AndroidJavaObject GetNativeAdManager()
    {
        if (mNativeAdManager != null)
        {
            return mNativeAdManager;
        }
        if (activity == null)
        {
            var unityPlayer = new AndroidJavaClass(
            "com.unity3d.player.UnityPlayer");
            activity = unityPlayer.GetStatic<AndroidJavaObject>(
           "currentActivity");
        }
        var jc = new AndroidJavaClass(
                    "com.bytedance.android.NativeAdManager");
        mNativeAdManager = jc.CallStatic<AndroidJavaObject>
("getNativeAdManager");
        return mNativeAdManager;
    }
```

2.4.3 Description of Custom Ad View Based on Android Platform

Java classes need to be introduced in the Unity project to manage and customize the view. The following key classes in the examples we provide are available for reference.

Native ad management：com.bytedance.android.NativeAdManager.java
Data binding logic of the native ads, the ad registration logic, the display of the native banner and the interstitial ads, the developer can customize the implementation.

Custom native interstitial ad：com.bytedance.android.IntersititialView.java Customization of interstitial ads. This class simply provides a layout style, and the developer needs to customize the layout according to his own UI style.

Custom native banner ad：com.bytedance.android.BannerView.java Customization of banner ads. This class simply provides a layout style, and developers need to customize the layout according to their own UI style.

For detailed code, please refer to the demo, pay attention to the notes in the custom layout.**After integration, you need to run enough tests to ensure normal ad billing.** If you have any questions, please contact us for support.

2.4.4 Add Dislike Logic to Native Ad

Add Dislike logic on Android platform
Adding dislike logic facilitates better matching and conversion rates of ad serving. Developers can refer to the example of adding dislike in com.bytedance.android.NativeAdManager.java: that is mentioned above

```
//Add dislike logic to the Pangle Ad Network helps to improve the accuracy of
ad delivery
    private void bindDislikeAction(TTNativeAd ad, View dislikeView,
TTAdDislike.DislikeInteractionCallback callback ) {
        final TTAdDislike ttAdDislike = ad.getDislikeDialog((Activity)
mContext);
        if (ttAdDislike != null) {
            ttAdDislike.setDislikeInteractionCallback(callback);
        }
        dislikeView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (ttAdDislike != null)
                    ttAdDislike.showDislikeDialog();
            }
        });
    }
```

An example of the use in native ads can be found in Demo.

2.4.5 Native Ad Access for iOS

- iOS ad access and callbacks are the same in C# layer as Android. But the UI needs to be customized at `NativeAd.mm`,The demo uses `customview` as the container to customize the corresponding view.
- When you get the refreshed data, you need to register the corresponding ad view to notify the SDK to listen for clicks and show events.

  // Register UIView with the native ad; the whole UIView will be clickable.
  [nativeAd registerContainer:self.customview withClickableViews:@[self.infoLabel, self.actionButton]];

- Only supports one type of ad at the same time. If there are multiple types, you need to customize multiple load events.
- Callback events need to be given the corresponding Context

```
/**
 This method is called when native ad material loaded successfully.
 */
```

- (void)nativeAdDidLoad:(BUNativeAd *)nativeAd
  {
  self.onNativeAdLoad((__bridge void)self,self.loadContext);
  }

```
/**
 This method is called when native ad materia failed to load.
 @param error : the reason of error
 */
```

- (void)nativeAd:(BUNativeAd *)nativeAd didFailWithError:(NSError *_Nullable)error
  {
  self.onError((int)error.code, AutonomousStringCopy2([[error localizedDescription] UTF8String]), self.loadContext);
  }

```
/**
 This method is called when native ad slot has been shown.
 */
```

- (void)nativeAdDidBecomeVisible:(BUNativeAd *)nativeAd
  {
  self.onAdShow(self.interactionContext);
  }

```
/**
 This method is called when native ad is clicked.
 */
```

- (void)nativeAdDidClick:(BUNativeAd *)nativeAd withView:(UIView *_Nullable)view
  {
  self.onAdDidClick(self.interactionContext);
  }

```
/**
 This method is called when the user clicked dislike reasons.
 Only used for dislikeButton in BUNativeAdRelatedView.h
 @param filterWords : reasons for dislike
 */
```

- (void)nativeAd:(BUNativeAd *)nativeAd dislikeWithReason:(NSArray<BUDislikeWords *> *)filterWords
  {
  self.onAdClose(self.interactionContext);

```
    [self.customview removeFromSuperview];
}
```

2.5 Rewarded Video Ad

The SDK provides rewarded video ad for the access party, and the user will be issued a reward upon completion of the video ad, usage scenarios include but are not limited to:

- 1.Watch video ad in games and get in-game gold coins, etc.: users must watch the full video to get rewards;
- 2.Points apps;
- Supported ad sizes: support full-screen display with horizontal screen and vertical screen, the default setting is horizontal screen

2.5.1 RewardVideoAd Interface

```
public sealed class RewardVideoAd : IDisposable
   {
       /// <inheritdoc/>
       public void Dispose()
       {
       }

       /// <summary>
       /// Set up ad interaction listener.
       /// </summary>
       public void SetRewardAdInteractionListener(
           IRewardAdInteractionListener listener)
       {
       }

       /// <summary>
       /// Set up ad download listener (Android).
       /// </summary>
       public void SetDownloadListener(IAppDownloadListener listener)
       {
       }

       /// <summary>
       /// Get the ad interaction type.
       /// </summary>
       public int GetInteractionType()
       {
           return 0;
       }

       /// <summary>
       /// Show rewarded video ads.
```

```
        /// </summary>
        public void ShowRewardVideoAd()
        {
        }

        /// <summary>
        /// Set whether to show the download bar.
        /// </summary>
        public void SetShowDownLoadBar(bool show)
        {
        }
    }
```

2.5.2 Load Rewarded Video Ad

Since it takes time for the rewarded video ad to load, it is recommended that you load the ad ahead of time and set up the listener for the callback event.

```
public void LoadRewardAd()
{
    if (this.rewardAd != null)
    {
        Debug.LogError("Ads already loaded");
        this.information.text = "Ads already loaded";
        return;
    }

    var adSlot = new AdSlot.Builder()
#if UNITY_IOS
        .SetCodeId("900546826")
#else
        .SetCodeId("901121430")
#endif
        .SetSupportDeepLink(true)
        .SetImageAcceptedSize(1080, 1920)
        .SetRewardName("Gold Coin") // Name of the reward
        .SetRewardAmount(3) // Number of rewards
        .SetUserID("user123") // User id, required parameter
        .SetMediaExtra("media_extra") // Additional parameters, optional
        .SetOrientation(AdOrientation.Horizontal) // Required parameter, set
how you wish the video ad to be displayed, horizontal or vertical
        .Build();

    this.AdNative.LoadRewardVideoAd(
        adSlot, new RewardVideoAdListener(this));
}
```

Ad loading completion listener

It is recommended to display ad after `OnRewardVideoCached` for better ad performance.

```
private sealed class RewardVideoAdListener : IRewardVideoAdListener
{
    private Example example;

    public RewardVideoAdListener(Example example)
    {
        this.example = example;
    }

    // Load rewarded video ad falied
    public void OnError(int code, string message)
    {
        Debug.LogError("OnRewardError: " + message);
        this.example.information.text = "OnRewardError: " + message;
    }

    // Load rewarded video ad successful, indicating that it can be played
online without local video cache
    public void OnRewardVideoAdLoad(RewardVideoAd ad)
    {
        Debug.Log("OnRewardVideoAdLoad");
        this.example.information.text = "OnRewardVideoAdLoad";

        ad.SetRewardAdInteractionListener(
            new RewardAdInteractionListener(this.example));
        ad.SetDownloadListener(
            new AppDownloadListener(this.example));

        this.example.rewardAd = ad;
    }

    // Successfully cached video data, indicating that it can be played using
the local cache.
    public void OnRewardVideoCached()
    {
        Debug.Log("OnRewardVideoCached");
        this.example.information.text = "OnRewardVideoCached";
    }
}
```

2.5.3 Display Rewarded Video Ad

Call this method ShowRewardVideoAd after ad completes to load.

```
public void ShowRewardAd()
{
    if (this.rewardAd == null)
    {
        Debug.LogError("Please load the ad first");
        this.information.text = "Please load the ad first";
        return;
    }
    this.rewardAd.ShowRewardVideoAd();
}
```

2.5.4 Monitoring Settings of Rewarded Video Ad

`RewardAdInteractionListener` Ad clicks and impressions related

```
private sealed class RewardAdInteractionListener :
IRewardAdInteractionListener
{
    private Example example;

    public RewardAdInteractionListener(Example example)
    {
        this.example = example;
    }
    // ad display
    public void OnAdShow()
    {
        Debug.Log("rewardVideoAd show");
        this.example.information.text = "rewardVideoAd show";
    }
    // The video is clicked
    public void OnAdVideoBarClick()
    {
        Debug.Log("rewardVideoAd bar click");
        this.example.information.text = "rewardVideoAd bar click";
    }
    // The ad is closed
    public void OnAdClose()
    {
        Debug.Log("rewardVideoAd close");
        this.example.information.text = "rewardVideoAd close";
    }
    // video playback is complete
    public void OnVideoComplete()
    {
        Debug.Log("rewardVideoAd complete");
        this.example.information.text = "rewardVideoAd complete";
    }
```

```
    // video playback error
    public void OnVideoError()
    {
        Debug.LogError("rewardVideoAd error");
        this.example.information.text = "rewardVideoAd error";
    }

    // The reward is valid (requires backend configuration regarding whether
to issue a reward)
    public void OnRewardVerify(
        bool rewardVerify, int rewardAmount, string rewardName)
    {
        Debug.Log("verify:" + rewardVerify + " amount:" + rewardAmount +
            " name:" + rewardName);
        this.example.information.text =
            "verify:" + rewardVerify + " amount:" + rewardAmount +
            " name:" + rewardName;
    }
}
```

`AppDownloadListener` Ad download related (only Android has a corresponding callback, iOS has no internal download)

```
private sealed class AppDownloadListener : IAppDownloadListener
{
    private Example example;

    public AppDownloadListener(Example example)
    {
        this.example = example;
    }

    public void OnIdle()
    {
    }

    public void OnDownloadActive(
        long totalBytes, long currBytes, string fileName, string appName)
    {
        Debug.Log("Downloading, click on the download area to pause");
        this.example.information.text = "Downloading, click on the download
area to pause";
    }

    public void OnDownloadPaused(
        long totalBytes, long currBytes, string fileName, string appName)
```

```
    {
        Debug.Log("Download paused, click download area to continue");
        this.example.information.text = "Download paused, click download area
to continue";
    }

    public void OnDownloadFailed(
        long totalBytes, long currBytes, string fileName, string appName)
    {
        Debug.LogError("Download failed, click download area to re-
        download");
        this.example.information.text = "Download failed, click download area
to re-
        download";
    }

    public void OnDownloadFinished(
        long totalBytes, string fileName, string appName)
    {
        Debug.Log("Download completed, click download area to re-
        download");
        this.example.information.text = "Download completed, click download
area to re-
        download";
    }

    public void OnInstalled(string fileName, string appName)
    {
        Debug.Log("Installation is complete, click on the download area to
open");
        this.example.information.text = "Installation is complete, click on
the download area to open";
    }
}
```

2.5.5 Rewarded Video Ad Destruction

Since the rewarded video ad is always referenced, the corresponding resource will not be released and needs to be released manually. `RewardVideoAd`

```
public void DisposeAds()
{
#if UNITY_IOS
    if (this.rewardAd != null)
    {
        this.rewardAd.Dispose();
        this.rewardAd = null;
```

```
    }
#else
    if (this.rewardAd != null)
    {
        this.rewardAd = null;
    }
#endif
}
```

If you need more than one rewarded video ad, it is recommended to create multiple `RewardVideoAd` and manage the corresponding video separately.
If only one is displayed at a time, `LoadRewardAd` can be called after the first rewarded video is closed to load the new rewarded video ad. Please refer to the demo example for specific usage.

2.6 Full-Screen Video Ad

The SDK provides full-screen video ad for the access party, the ad plays in full-screen and can be skipped after a certain time, watching the whole video is not required. Supported ad sizes: support full-screen display horizontally or vertically, the default setting is to play horizontally.

2.6.1 FullScreenVideoAd Interface

```
public sealed class FullScreenVideoAd
   {
        public void Dispose()
       {
       }
       /// <summary>
       /// Set the ad interaction event listener
       /// </summary>
       public void SetFullScreenVideoAdInteractionListener(
           IFullScreenVideoAdInteractionListener listener)
       {
       }

       /// <summary>
       /// Set app install ad listener (Android)
       /// </summary>
       public void SetDownloadListener(IAppDownloadListener listener)
       {
       }

       /// <summary>
       /// Get the interaction type for the ad
       /// </summary>
       public int GetInteractionType()
       {
```

```
            return 0;
        }

        /// <summary>
        /// Display full-screen video ads
        /// </summary>
        public void ShowFullScreenVideoAd()
        {
        }

        /// <summary>
        /// Set whether to show the current bar
        /// </summary>
        public void SetShowDownLoadBar(bool show)
        {
        }
    }
```

2.6.2 Load Full-Screen Video Ad

Since it takes time to load a full-screen video ad, it is recommended that you load the ad ahead of time and set up the listener for the callback event.

```
 public void LoadFullScreenVideoAd()
    {
        var adSlot = new AdSlot.Builder()
#if UNITY_IOS
        .SetCodeId("900546299")
#else
        .SetCodeId("901121375")
#endif
        .SetSupportDeepLink(true)
        .SetImageAcceptedSize(1080, 1920)
        .SetOrientation(AdOrientation.Horizontal)
        .Build();
        this.AdNative.LoadFullScreenVideoAd(adSlot, new
FullScreenVideoAdListener(this));

    }
```

Ad loading completion listener

It is recommended to display ad after `OnFullScreenVideoCached` for better ad performance.

```
private sealed class FullScreenVideoAdListener : IFullScreenVideoAdListener
    {
        private Example example;

        public FullScreenVideoAdListener(Example example)
```

```
        {
            this.example = example;
        }
        // Failed to load full-screen video ad
        public void OnError(int code, string message)
        {
            Debug.LogError("OnFullScreenError: " + message);
            this.example.information.text = "OnFullScreenError: " + message;
        }
        // Load full-screen video ad successfully, indicating that it can be
played online without local video cache
        public void OnFullScreenVideoAdLoad(FullScreenVideoAd ad)
        {
            Debug.Log("OnFullScreenAdLoad");
            this.example.information.text = "OnFullScreenAdLoad";

            ad.SetFullScreenVideoAdInteractionListener(
                new FullScreenAdInteractionListener(this.example));
            ad.SetDownloadListener(
                new AppDownloadListener(this.example));

            this.example.fullScreenVideoAd = ad;
        }
        //Successfully cache video data, indicating that you can use the local
cache to play
        public void OnFullScreenVideoCached()
        {
            Debug.Log("OnFullScreenVideoCached");
            this.example.information.text = "OnFullScreenVideoCached";
        }
    }
```

2.6.3 Display Full-Screen Video Ad

Call this method ShowFullScreenVideoAd after ad completes to load.

```
public void ShowFullScreenVideoAd()
{
    if (this.fullScreenVideoAd == null)
    {
        Debug.LogError("Please load the ad first");
        this.information.text = "Please load the ad first";
        return;
    }
    this.fullScreenVideoAd.ShowFullScreenVideoAd();
}
```

2.6.4 Monitoring Settings of Full-Screen Video Ad

`FullScreenAdInteractionListener` Ad clicks and impressions related

```
private sealed class FullScreenAdInteractionListener :
IFullScreenVideoAdInteractionListener
    {
        private Example example;

        public FullScreenAdInteractionListener(Example example)
        {
            this.example = example;
        }

        public void OnAdShow()
        {
            Debug.Log("fullScreenVideoAd show");
            this.example.information.text = "fullScreenVideoAd show";
        }

        public void OnAdVideoBarClick()
        {
            Debug.Log("fullScreenVideoAd bar click");
            this.example.information.text = "fullScreenVideoAd bar click";
        }

        public void OnAdClose()
        {
            Debug.Log("fullScreenVideoAd close");
            this.example.information.text = "fullScreenVideoAd close";
        }

        public void OnVideoComplete()
        {
            Debug.Log("fullScreenVideoAd complete");
            this.example.information.text = "fullScreenVideoAd complete";
        }

        public void OnVideoError()
        {
            Debug.Log("fullScreenVideoAd OnVideoError");
            this.example.information.text = "fullScreenVideoAd OnVideoError";
        }

        public void OnSkippedVideo()
        {
            Debug.Log("fullScreenVideoAd OnSkippedVideo");
            this.example.information.text = "fullScreenVideoAd skipped";

        }
    }
```

`AppDownloadListener`
Advertising download related (only Android has a corresponding callback, iOS has no internal download), the specific use is consistent with rewarded video ad.

2.6.5 Full-Screen Video Ad Destruction

Since full-screen video ad is always referenced, the corresponding resources `FullScreenVideoAd` will not be released and need to be released manually.

```
public void DisposeAds()
{
#if UNITY_IOS
    if (this.fullScreenVideoAd != null)
    {
        this.fullScreenVideoAd.Dispose();
        this.fullScreenVideoAd = null;
    }
#else
    if (this.fullScreenVideoAd != null)
    {
        this.fullScreenVideoAd = null;
    }
#endif
}
```

If you need more than one full-screen video ads, it is recommended to create multiple `FullScreenVideoAd` to manage the corresponding video separately.
If only one is displayed at a time, `LoadFullScreenVideoAd` can be called after the first full-screen video ad is closed to load the new full-screen video ad. Please refer to the demo example for specific usage.

2.7 Customized Express Ads

2.7.1 Customized express stream ads

- **Description:** Customized express stream ads are native ads with dynamic rendering capabilities, which means that developers can edit the rendering template on the media platform. The SDK supports real-time update of the ad layout. It also features a rendering function and provides rendering views for developers.
- **Instructions: iOS**
- You can configure basic information of customized express stream ads by using BUNativeExpressAdManager, for example expected size. When configuring the desired size, in order to avoid deformation of the ad view during the rendering process, it is recommended to use the same size configurations as the media platform. You can also

configure the number of ads required, with a maximum of three at a time. By setting BUNativeExpressAdViewDelegate, you can get callbacks for ads, impressions, clicks, and closes. Developers can get the displayed ad view through BUNativeExpressAdView, in which the render method should be invoked to trigger rendering of the ad view. Rendering will be triggered after the ad material is obtained. Refer to the demo for details. The isReady method can be used to query whether the view was rendered successfully. Please note that the rootViewController must be set as this is the viewController required for jumping to the landing page.

- **Instructions：Android**
- 1.You can configure the desired size of customized express stream ads by using SetExpressViewAcceptedSize of the AdSlot.
- 2.Invoke LoadNativeExpressAd(AdSlot adSlot, IExpressAdListener listener) of the AdNative to load the ad object ExpressAd.
- 3.Invoke the NativeAdManager.Instance().ShowExpressFeedAd method to display ads. Refer to the demo for a detailed example.
- 4. Note that after using the ad object, you need to invoke NativeAdManager.Instance().DestoryExpressAd to release resources and reduce memory usage.

public void DisposeAds()
{
public sealed class ExpressAd {

```
    public int index;

    internal ExpressAd (AndroidJavaObject expressAd) { }
    /// <summary>
    /// Gets the interaction type.
    /// </summary>
    public int GetInteractionType () {
        return 0;
    }
    public AndroidJavaObject handle { get { return null; } }

    /// <summary>
    /// Sets the interaction listener for this Ad.
    /// </summary>
    public void SetExpressInteractionListener
(IExpressAdInteractionListener listener)
    {
    }

    /// <summary>
    /// Sets the dislike callback.
    /// </summary>
    /// <param name="dislikeCallback">Dislike callback.</param>
```

```
    public void SetDislikeCallback(IDislikeInteractionListener
dislikeCallback)
    {

    }

    /// <summary>
    /// Sets the download listener.
    /// </summary>
    public void SetDownloadListener (IAppDownloadListener listener) { }

    /// <summary>
    /// Set this Ad not allow sdk count down.
    /// </summary>
    public void SetNotAllowSdkCountdown () { }

    /// <summary>
    /// show the  express Ad
    /// <param name="x">the x of th ad</param>
    /// <param name="y">the y of th ad</param>
    /// </summary>
    public void ShowExpressAd (float x, float y) { }

    /// <inheritdoc/>
    public void Dispose() { }

    /// <summary>
    /// Sets the slide interval time.
    /// </summary>
    /// <param name="intervalTime">Interval time.</param>
    public void SetSlideIntervalTime(int intervalTime){}

}
```

2.7.2 Customized Express Banner Ads

- **Description：** Customized express banner ads are native ads with dynamic rendering capabilities, which means that developers can edit the rendering template on the media platform. The SDK supports real-time update of the ad layout. It also features a rendering function and provides rendering views for developers.
- **Instructions：iOS**
- You can configure basic information of customized express banner ads by using BUNativeExpressBannerView. For example, when configuring the desired size, in order to avoid deformation of the ad view during the rendering process, you must use the same size configurations as the media platform. By setting BUNativeExpressBannerViewDelegate, you can get callbacks for ads, impressions, clicks, and closes. Please note that the rootViewController must be set as this is the viewController required for jumping to the landing page.

- **Instructions：Android**
- 1.You can configure the desired size of customized express banner ads by using SetExpressViewAcceptedSize of the AdSlot.
- 2.Invoke LoadExpressBannerAd(AdSlot adSlot, IExpressAdListener listener) of the AdNative to load the ad object ExpressAd.
- 3.Invoke the NativeAdManager.Instance().ShowExpressBannerAd method to display ads. Refer to the demo for a detailed example.
- 4.Note that after using the ad object, you need to invoke NativeAdManager.Instance().DestoryExpressAd to release resources and reduce memory usage.

```
public sealed class ExpressBannerAd {

    public int index;

    internal ExpressBannerAd(AndroidJavaObject expressAd) { }
    /// <summary>
    /// Gets the interaction type.
    /// </summary>
    public int GetInteractionType () {
        return 0;
    }
    public AndroidJavaObject handle { get { return null; } }

    /// <summary>
    /// Sets the interaction listener for this Ad.
    /// </summary>
    public void SetExpressInteractionListener
(IExpressAdInteractionListener listener)
    {
    }

    /// <summary>
    /// Sets the dislike callback.
    /// </summary>
    /// <param name="dislikeCallback">Dislike callback.</param>
    public void SetDislikeCallback(IDislikeInteractionListener
dislikeCallback)
    {

    }

    /// <summary>
    /// Sets the download listener.
    /// </summary>
    public void SetDownloadListener (IAppDownloadListener listener) { }
```

```
    /// <summary>
    /// Set this Ad not allow sdk count down.
    /// </summary>
    public void SetNotAllowSdkCountdown () { }

    /// <summary>
    /// show the  express Ad
    /// <param name="x">the x of th ad</param>
    /// <param name="y">the y of th ad</param>
    /// </summary>
    public void ShowExpressAd (float x, float y) { }

    /// <inheritdoc/>
    public void Dispose() { }

}
```

2.7.3 Customized Express Interstitial Ads

- **Description：** Customized express interstitial ads are native ads with dynamic rendering capabilities, which means that developers can edit the rendering template on the media platform. The SDK supports real-time update of the ad layout. It also features a rendering function and provides rendering views for developers.
- **Instructions：iOS** You can configure basic information of customized express interstitial ads by using BUNativeExpressInterstitialAd. For example, when configuring the desired size, in order to avoid deformation of the ad view during the rendering process, you must use the same size configurations as the media platform. By setting BUNativeExpresInterstitialAdDelegate, you can get callbacks for ads, impressions, clicks, and closes. Please note that the rootViewController must be set as this is the viewController required for jumping to the landing page.
- **Instructions：Android**
- 1.You can configure the desired size of customized express interstitial ads by using SetExpressViewAcceptedSize of the AdSlot.
- 2.Invoke LoadExpressInterstitialAd(AdSlot adSlot, IExpressAdListener listener) of the AdNative to load the ad object ExpressAd.
- 3.Invoke the NativeAdManager.Instance().ShowExpressInterstitialAd method to display ads. Refer to the demo for a detailed example.
- 4.Note that after using the ad object, you need to invoke NativeAdManager.Instance().DestoryExpressAd to release resources and reduce memory usage.

```
/// <summary>
/// The express Ad.
/// </summary>
public sealed class ExpressInterstitialAd {
```

```
    public int index;

    internal ExpressInterstitialAd(AndroidJavaObject expressAd) { }
    /// <summary>
    /// Gets the interaction type.
    /// </summary>
    public int GetInteractionType () {
        return 0;
    }
    public AndroidJavaObject handle { get { return null; } }

    /// <summary>
    /// Sets the interaction listener for this Ad.
    /// </summary>
    public void SetExpressInteractionListener
(IExpressAdInteractionListener listener)
    {
    }

    /// <summary>
    /// Sets the dislike callback.
    /// </summary>
    /// <param name="dislikeCallback">Dislike callback.</param>
    public void SetDislikeCallback(IDislikeInteractionListener
dislikeCallback)
    {

    }

    /// <summary>
    /// Sets the download listener.
    /// </summary>
    public void SetDownloadListener (IAppDownloadListener listener) { }

    /// <summary>
    /// Set this Ad not allow sdk count down.
    /// </summary>
    public void SetNotAllowSdkCountdown () { }

    /// <summary>
    /// show the  express Ad
    /// <param name="x">the x of th ad</param>
    /// <param name="y">the y of th ad</param>
    /// </summary>
    public void ShowExpressAd (float x, float y) { }

    /// <inheritdoc/>
    public void Dispose() { }
```

```
}
```

2.7.4 Customized Express Rewarded Video Ads

- **Description：** Customized express rewarded video ads are native ads with dynamic rendering capabilities, which means that developers can edit the rendering template on the media platform. The SDK supports real-time update of the ad layout. It also features a rendering function and provides rendering views for developers.
- **Instructions：**
- iOS:You can request customized express rewarded video ads by using BUNativeExpressRewardedVideoAd. Invoke showAdFromRootViewController to display ads. Please note that the rootViewController must be set as this is the viewController required for displaying the ad and jumping to the landing page. By setting BUNativeExpressRewardedVideoAdDelegate, you can get callbacks for ads, impressions, clicks, and closes.
- To ensure smooth playback and display, it is recommended to display the video only after receiving the callback of successful rendering and video download.
- Android: The method is the same as that for ordinary rewarded video ads. Please note that you need to use the applied code bit related to the customized template.

```
/// <summary>
/// The reward video Ad.
/// </summary>
public sealed class ExpressRewardVideoAd : IDisposable
{
    /// <inheritdoc/>
    public void Dispose()
    {
    }
    /// <summary>
    /// Sets the interaction listener for this Ad.
    /// </summary>
    public void SetRewardAdInteractionListener(
        IRewardAdInteractionListener listener)
    {
    }
    /// <summary>
    /// Sets the download listener.
    /// </summary>
    public void SetDownloadListener(IAppDownloadListener listener)
    {
    }
    /// <summary>
    /// Gets the interaction type.
    /// </summary>
    public int GetInteractionType()
    {
```

```
        return 0;
    }
    /// <summary>
    /// Show the reward video Ad.
    /// </summary>
    public void ShowRewardVideoAd()
    {
    }
    /// <summary>
    /// Sets whether to show the download bar.
    /// </summary>
    public void SetShowDownLoadBar(bool show)
    {
    }
}
```

2.7.5 Customized Express Full-screen Video Ads

- **Description**： Customized express full-screen video ads are native ads with dynamic rendering capabilities, which means that developers can edit the rendering template on the media platform. The SDK supports real-time update of the ad layout. It also features a rendering function and provides rendering views for developers.
- **Instructions**：
- iOS: You can request customized express full-screen video ads by using BUNativeExpressFullscreenVideoAd. Invoke showAdFromRootViewController to display ads. Please note that the rootViewController must be set as this is the viewController required for displaying the ad and jumping to the landing page. By setting BUNativeExpressFullscreenVideoAdDelegate, you can get callbacks for ads, impressions, clicks, and closes.
- To ensure smooth playback and display, it is recommended to display the video only after receiving the callback of successful rendering and video download.
- Android: The method is the same as that for ordinary full-screen video ads. Please note that you need to use the applied code bit related to the customized template.

```
/// <summary>
/// The full screen video Ad.
/// </summary>
public sealed class ExpressFullScreenVideoAd
{
     public void Dispose()
    {
    }
    /// <summary>
    /// Sets the interaction listener for this Ad.
    /// </summary>
```

```
    public void SetFullScreenVideoAdInteractionListener(
        IFullScreenVideoAdInteractionListener listener)
    {
    }
    /// <summary>
    /// Sets the listener for the Ad download.
    /// </summary>
    public void SetDownloadListener(IAppDownloadListener listener)
    {
    }
    /// <summary>
    /// Gets the interaction type.
    /// </summary>
    public int GetInteractionType()
    {
        return 0;
    }
    /// <summary>
    /// Show the full screen video.
    /// </summary>
    public void ShowFullScreenVideoAd()
    {
    }
    /// <summary>
    /// Set to show the download bar.
    /// </summary>
    public void SetShowDownLoadBar(bool show)
    {
    }
}
```

3.Reference

3.1 SDK Error Code

The following is a description of the relevant error code. Please contact us if you have any other questions.

Error Code	Description
20000	Success
40000	http content type error
40001	http request pb error

40002	source_type='app', app initiating request can't be empty
40003	source_type='wap', wap initiating request cannot be empty
40004	Ad slot cannot be empty
40005	Ad slot size cannot be empty
40006	Invalid ad slot ID
40007	Incorrect number of ads
40008	Image size error
40009	Media ID is invalid
40010	Media type is invalid
40011	Ad type is invalid
40012	Media access type is illegal (deprecated)
40013	Code bit id is less than 900000000, but adType is not splash
40014	The redirect parameter is incorrect
40015	Media rectification exceeds deadline, request illegal
40016	The relationship between slot_id and app_id is invalid.
40017	Media access type is not valid API/SDK
40018	Media package name is inconsistent with entry
40019	Media configuration adtype and request are inconsistent
40020	The ad space registered by developers exceeds daily request limit
40021	Apk signature sha1 value is inconsistent with media platform entry
40022	Whether the media request material is inconsistent with the media platform entry
40023	The os field is incorrect
40024	Sdk version is too low to return ads
50001	Server Error
60001	Show event processing error
60002	Click event processing error
60007	Server abnormity or failure of rewarded video ad rewards verification
-1	Data parsing failed

-2	Network Error
-3	Parsing data without ad
-4	Return data is missing the necessary fields
-5	bannerAd image failed to load
-6	Interstitial ad image failed to load
-7	Splash ad image failed to load
-8	Frequent request
-9	Request entity is empty
-10	Cache parsing failed
-11	Cache expired
-12	No splash ad in the cache
101	Rendering result data parsing failed
102	Master template is invalid
103	Template difference is invalid
104	Material data is abnormal
105	Template data parsing is abnormal
106	Rendering is abnormal
107	Render recall timeout