YUMIMOBI Ad SDK

Integration Guide

For Android Developers

V3.2.2

Contents

1. Overview

1.1. Target Readers

This document is for Android developers who want to integrate YUMIMOBI advertising SDK into their product.

1.2. Development Environment

OS: Windows， Mac， Linux

Android SDK: 2.3 (API level 9)

IDE: Eclipse with ADT (ADT version 23.0.4)，Android-studio

Java: jdk7

2. Development Environment Configuration

2.1. Add lib file

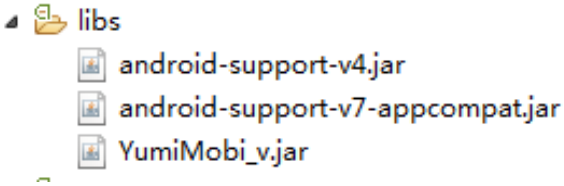
Android-studio:

```
// ensure whether jcenter is supported in build.gradle under Project root directory of android studio
buildscript {
   repositories {
      jcenter()
   }
}
allprojets {
   repositories {
      jcenter()
   }
}
//Add dependency in module build. Gradle
dependencies {
      compile 'com.yumimobi.ads:mediation:3.2.2.+'
 }
```

Eclipse:

All lib files are placed in lib in the SDK:

- YumiMobi_vX.X.X.jar
- android-support-v4.jar
- android-support-v7-appcompat.jar
- lib project of google_play_service



Create libs folder under the root directory of your project, you can choose to or not to add android-support-v4.jar and/or android-support-v7-appcompat.jar into libs according to your needs. You must use the jar file provided by YUMIMOBI when you need to use v4.jar or v7.jar.

About google_play_service project:

google_play_service is not mandatory, while some ad platforms need it. YUMIMOBI does not need google_play_service. You need use it as a library and import it into your project. Also, add the ollowing code in tab <application> of your manifest.xml.

```
<meta-data
  android:name="com.google.android.gms.version"
  class="kix-line-break"
  android:value="@integer/google_play_services_version" />
```

2.2. Add permission

Add the following permissions in manifest.xml of your project:

Mandatory:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

```
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>

<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />

<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
```

Optional:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>

<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.DOWNLOAD_WITHOUT_NOTIFICATION" />
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.SEND_SMS"/>

<uses-permission android:name="android.permission.WRITE_CALENDAR"/>
```

2.3. Registered components

Add following in manifest.xml of your project:

```
<receiver android:name="com.yumi.android.sdk.ads.self.module.receiver.ADReceiver" >
    <intent-filter>
        <action android:name="android.intent.action.DOWNLOAD_COMPLETE" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.PACKAGE_ADDED" />
        <data android:scheme="package" />
    </intent-filter>
</receiver>
<service   android:name="com.yumi.android.sdk.ads.service.YumiAdsEventService" />
<activity   android:name="com.yumi.android.sdk.ads.self.activity.YumiFullScreenActivity"
            android:configChanges="keyboardHidden|orientation|screenSize"
            android:theme="@android:style/Theme.NoTitleBar.Fullscreen" />
<!—Debugging Activity -->

<activity android:name="com.yumi.android.sdk.ads.mediation.activity.MediationTestActivity" ></activity>
```

3 Integration

3.1 Banner

Creat a ViewGroup as banner container, and add it in Activity.

Then call the code below:

```
//create YumiBanner object. Activity is the activity which you need to display banner ad. You need to create a SlotID on YUMIMOBI. Auto indicates if the mode is automatic.
//auto==true    Banner ads automatic rotation
//auto==flase    Banner ads manual rotation，call banner.requestYumiBanner() repeatedly to rotate
// If you are using YUMI mediation alone, please enable YUMI ad rotation, set the field as “true”; if you are using YUMI ads in other mediations, to ensure ad performance, please disable YUMI ad rotation, set the field as “false”.
banner = new YumiBanner(activity，  "YOUR_SLOT_ID"，  auto);
//Set ViewGroup as banner container, set it along with size
// bannerContainer    Your ad container
// AdSize.BANNER_SIZE_AUTO    SDK automatically sets screen size as 320*50 or 728*90
// isMatchWindowWidth ==true Banner width equals screen width
banner.setBannerContainer(bannerContainer，  AdSize.BANNER_SIZE_AUTO，  isMatchWindowWidth);
//Set channel according to your settings on the platform, you only need to set it once. Repeated calls are based on the last time you call.
banner.setChannelID(channelStr);
// Set version name according to your settings on the platform, you only need to set it once. Repeated calls are based on the last time you call.
banner.setVersionName(versionStr);
//Start requesting ads, auto==true    The method needs to be called only oncebanner.requestYumiBanner();
```

Note: ChannelID refers to the channel labeling of the application, and the YUMI platform can carry out data statistics and effect analysis according to the ChannelID. A Popstar! For example, when the game is released to the SamSung channel, setChannelID(channelStr) needs to be set to setChannelID(' SamSung ').

The channelID is labeled as the YUMI platform to generate the information and cannot be modified at will：

Channel name	**ChannelID**
SamSung	SamSung

Implement in Activity lifecycle:

```
@Override
protected void onDestroy() {
        if (banner!= null) {
```

```
                banner.onDestory();
            }
            super.onDestroy();
        }
```

3.2 Interstitial

Add the following code in onCreate method of Activity:

```
//Create YumiInterstitial object. Activity is the one you use to show interstitials, You need to create a SlotID on YUMIMOBI.
 Auto indicates if the mode is automatic.
//auto==true    Automatically request the next ad, auto mode recommended to ensure ad performance
//auto==flase    Auto request disabled, to request please repeatedly call interstitial.requestYumiInterstitial()
// If you are using YUMI Ads, please enable YUMI ad rotation, set the field as “true”.
interstitial = new YumiInterstitial(activity ,  "YOUR_SLOT_ID" ,  auto);
// Set channel according to your settings on the platform, you only need to set it once. Repeated calls are based on the last
time you call.
interstitial.setChannelID(channelStr);
// Set version name according to your settings on the platform, you only need to set it once. Repeated calls are based on the
last time you call.
interstitial.setVersionName(versionStr);
//Start requesting ads
interstitial.requestYumiInterstitial();
```

Note: ChannelID refers to the channel labeling of the application, and the YUMI platform can carry out data statistics and effect analysis according to the ChannelID. A Popstar! For example, when the game is released to the SamSung channel, setChannelID(channelStr) needs to be set to setChannelID(' SamSung ').

The channelID is labeled as the YUMI platform to generate the information and cannot be modified at will：

Channel name	**ChannelID**
SamSung	SamSung

Call the following code when you need to show interstitial ads:

```
//Show immediately, if there is a pre-cached interstitial, immediately show the pop-up interstitial (Speed to pop up varies on different devices, there might be visual lag), if there isn't a pre-cached interstitial, show no ads and drop the impression opportunity until the next call.
if (interstitial != null) {
        interstitial.showInterstitial(false);
}
//Delayed display    If when you intend to show interstitial, it's still pre-caching, then delayed display allows SDK to wait for the pre-cach to complete, after completion it will show interstitial. Time waited is indefinite.
if (interstitial != null) {
        interstitial.showInterstitial(true);
}
//Cancel delayed display    If you need to do other operations when delayed display hasn't started, you need to call the following method to cancel this delayed display. After cancellation, interstitial won't be shown until the next call.
if (interstitial != null) {
        interstitial.cancelInterstitialDelayShown();
}
```

Implement in Activity lifecycle:

```
    @Override
    protected void onDestroy() {
            if (interstitial != null) {
                    interstitial .onDestory();
            }
            super.onDestroy();
    }
```

As different platforms have different pop-up displays, you need to add the following code in **onBackPressed()**of Activity:

Note: In order not to confuse the logic of back key, please make sure to add this method when using interstitial

```
// please make sure you add this method in order to avoid confusing back key logic.
if (interstitial.onBackPressed()) {
        return ;
}
super.onBackPressed();
```

3.3 Rewarded Video

Add following code in onCreate method of Activity:

```
// Create YumiInterstitial object. Activity is the one you use to show interstitials, SlotID is the app ID which is assigned to you by the platform, auto indicates if the mode is automatic.
media = new YumiMedia(activity ,  "YOUR_SLOT_ID");
// Set channel according to your settings on the platform, you only need to set it once. Repeated calls are based on the last time you call.
media.setChannelID(channelStr);
// Set version name according to your settings on the platform, you only need to set it once. Repeated calls are based on the last time you call.
media.setVersionName(versionStr);
//Start requesting ads.
media.requestYumiMedia();
```

Note: For every YumiMedia instantiation, the method to start requesting ads needs to be called only once.

ChannelID refers to the channel labeling of the application, and the YUMI platform can carry out data statistics and effect analysis according to the ChannelID. A Popstar! For example, when the game is released to the SamSung channel, setChannelID(channelStr) needs to be set to setChannelID(' SamSung ').

The channelID is labeled as the YUMI platform to generate the information and cannot be modified at will：

Channel name	**ChannelID**
SamSung	SamSung

To check if video is available, call the following code:

```
if (media!= null) {
        media. isMediaPrepared ();
}
```

Note: It is recommended to request every five seconds

When you need to show rewarded video, call the following code:

```
//Show immediately
if (media!= null) {
        media.showMedia();
}
```

Note:

1. Rewarded video is available after the above integration , but reward callbacks are still unavailable. To get rewards callbacks, please add listener to get rewards callback according to status listener section in Advanced Features.
2. A new request for the next ad will be sent after the previous one has been closed or previous request has failed.
3. Method media.requestYumiMedia() needs to be called only once at the beginning.

Implement in Activity lifecycle:

```
    @Override
    protected void onDestroy() {
        if (media != null) {
            media.onDestory();
        }
        super.onDestroy();
    }
```

3.4 Splash

Add the following code in method onCreate of Activity:

```
//Create splash object.
```

```
//activity used to show splash
//SlotID AppID assigned by the platform
// container:ad container
// width/height:width and height of ad container
// SplashADListener:ad callback listener
splashAD = new SplashAD(activity， SlotID， container， adwidth， adheight， SplashADListener);
```

Call the following method in corresponding Activity lifecycle:

```
    @Override
    protected void onDestroy() {
        if (splashAD!= null) {
            splashAD.onDestory();
        }
        super.onDestroy();
    }
```

3.5 Native

Add the following code in method onCreate of Activity:

```
// Create a native ad, yid is Yumi ID for Yumi background
YumiNative nativeAd = new YumiNative(this, yid);
// set Channel ID
nativeAd.setChannelID(channelStr);
//set version
nativeAd.setVersionName(versionStr);
// set callback interface of native ad.
nativeAd.setNativeEventListener(new IYumiNativeListener()
{
    @Override
    public void onLayerPrepared(int adCount)
    {
        // callback of request success, adCount refers to returned ad quantity
    }
    @Override
    public void onLayerFailed(LayerErrorCode error)
    {
        // callback of request error, the error is notification of request failure
    }
```

```
    @Override
    public void onLayerClick() {
        // callback of  AD Click
    }
});
// request ad, the result of success or error will be returned in callback interface
nativeAd.requestYumiNative();
```

Note: ChannelID refers to the channel labeling of the application, and the YUMI platform can carry out data statistics and effect analysis according to the ChannelID. A Popstar! For example, when the game is released to the SamSung channel, setChannelID(channelStr) needs to be set to setChannelID(' SamSung ').

The channelID is labeled as the YUMI platform to generate the information and cannot be modified at will：

Channel name	ChannelID
SamSung	SamSung

Add corresponding SDK life-cycle method in onDestroy() of Activity lifecycle:

```
@Override
protected void onDestroy()
{
    super.onDestroy();
    if (nativeAd!=null)
    {
        nativeAd.onDestroy();
    }
}
```

4 Advanced Features

4.1 Banner

4.1.1 Set ad status listener

If you need to listen to the lifecycle of banner ads, please call the following method after you create YumiBanner object:

```
// Set ad status listener.
```

```
banner.setBannerEventListener(bannerListener);
```

Regarding ad listener, you can instantiate an IYumiBannerListener, and add your own logic according to the callback. The callbacks are shown below:

onBannerPreparedFailed(LayerErrorCode errorCode)	Callback when caching fails. Reasons can be found through errorCode.getMsg()
onBannerPrepared()	Callback when caching succeeds.
onBannerExposure()	Callback when impression succeeds.
onBannerClosed()	Callback when Banner is closed.
onBannerClicked()	Callback when Banner is clicked.

Sample

```
// Created banner ad status listener.
bannerListener = new IYumiBannerListener() {
                @Override
                public void onBannerPreparedFailed(LayerErrorCode errorCode) {
                        // Callback when caching fails. Reasons can be found through errorCode.getMsg()
                }
                @Override
                public void onBannerPrepared() {
                        //Callback when caching succeeds.
                }
                @Override
                public void onBannerExposure() {
                        //Callback when impression succeeds.
                }
                @Override
                public void onBannerClosed() {
                        //Callback when Banner is closed.
                }
                @Override
```

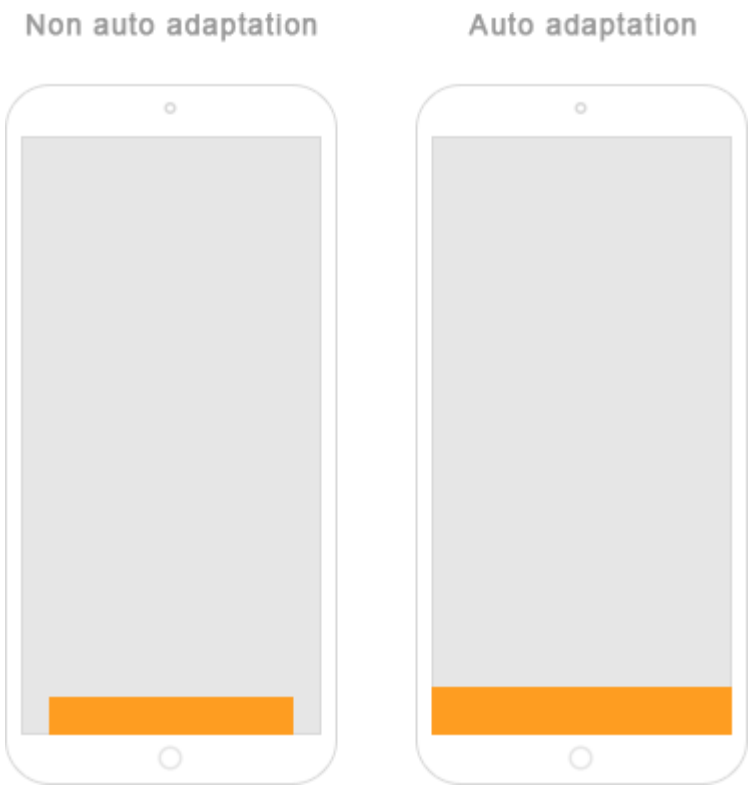
```
        public void onBannerClicked() {
            //Callback when Banner is clicked.
        }
    };
```

4.1.2 Show and hide banner

Use the following methods to show and hide banner:

```
//Hide banner, pause rotation at the same time.
banner.dismissBanner();
//Resume to show banner, resume rotation at the same time.
banner.resumeBanner();
```

4.1.3 Banner ads automatically adapt to screen size



When you set banner ad container, you can use isMatchWindowWidth, a parameter

of boolean type provided by YUMI SDK. This parameter indicates if banner width has populated full screen. When it's true, banner width equals the screen width.

4.2 Interstitial

Set ad status listener

If you need to listen to the lifecycle of interstitial ads, please call the following method after you create YumiBanner object:

```
// Set ad status listener.
interstitial.setInterstitialEventListener(interstitialListener);
```

Regarding ad listener, you can instantiate an IYumiInterstitialListener, and add your own logic according to the callback. The callbacks are shown below:

onInterstitialPreparedFailed(LayerErrorCode error)	Callback when caching fails. Reason can be found through errorCode.getMsg()
onInterstitialPrepared()	Callback when loading succeeds. Note: please do not call show interstitial method in this callback.
onInterstitialExposure()	Callback when impression succeeds.
onInterstitialClosed()	Callback when interstitial is closed.
onInterstitialClicked()	Callback when interstitial is clicked.

Sample

```
// Set ad status listener.
interstitialListener = new IYumiInterstititalListener() {
                @Override
                public void onInterstitialPreparedFailed(LayerErrorCode error) {
                        // Callback when loading fails. Reason can be found through errorCode.getMsg()
                }
                @Override
                public void onInterstitialPrepared() {
                        // Callback when loading succeeds.
```

```
        }
        @Override
        public void onInterstitialExposure() {
            // Callback when impression succeeds.
        }
        @Override
        public void onInterstitialClosed() {
            // Callback when interstitial is closed.
        }
        @Override
        public void onInterstitialClicked() {
            //Callback when interstitial is clicked.
        }
    };
```

4.3 Rewarded Video

Set ad status listener

If you need to listen to the lifecycle of video ads, please call the following method after you create YumiBanner object:

```
// Set ad status listener.
media.setMediaEventListner(mediaListener);
```

Regarding ad listener, you can instantiate an IYumiMediaListener, and add your own logic according to the callback. The callbacks are shown below:

onMediaExposure()	Callback when impression succeeds
onMediaClosed()	Callback when rewarded video is closed
onMediaClicked()	Callback when rewarded video is clicked. Note: this does not guarantee 100% callback due to platform differences.
onMediaIncentived()	Callback for rewards after rewarded video has been played completely. Note:

	if the video has not been played completely, this callback won't be used. In addition, this method is always triggered after onInterstitialClosed().

Sample

```
// Set ad status listener.
mediaListener = new IYumiMediaListener() {
                    @Override
                    public void onMediaIncentived() {
                        //Callback when reward is obtained
                    }
                    @Override
                    public void onMediaExposure() {
                        // Callback when impression succeeds.
                    }
                    @Override
                    public void onMediaClosed() {
                        //Callback when interstitial is closed.
                    }
                    @Override
                    public void onMediaClicked() {
                        //Callback when interstitial is clicked.
                    }
            };
```

4.4 Splash

Set ad status listener

Regarding ad listener, you can instantiate a SplashADListener, and add your own logic according to the callback. The callbacks are shown below:

onSplashShow()	Callback when splash is shown
onSplashClose()	Callback when splash is closed
onSplashClick()	Callback when splash is clicked
onSplashFailed()	Callback when loading fails

Sample

```
// Set ad status listener.
splashListener = new SplashADListener () {
                        @Override
                        public void onSplashShow () {
                                //Callback when splash is shown
                        }
                        @Override
                        public void onSplashFailed () {
                                //Callback when caching failed
                        }
                        @Override
                        public void onSplashClose () {
                                //Callback when splash is closed
                        }
                        @Override
                        public void onSplashClick () {
                                //Callback when splash is clicked
                        }
                };
```

4.5 Native

If you use native ads, please refer to the following method:

```
if (nativeAd.getADCount() > 0))// determine whether the next ad is available through the quantity of remaining ads
{
    final NativeContent content = nativeAd.nextContent();//call next ad
    int type = content.getContentType();//obtain ad types 1. material form 2. layout form
     content.getDesc();//obtain detail description
     content.getIcon_url();//obtain icon url
     content.getImg_url();//obtain image url
     content.getImg_height();//obtain the width of image, default 0 when unavailable
     content.getImg_width();//obtain the height of image, default 0 when unavailable
     content.getJumpUrl();//obtain click jump url
     content.getTitle();//obtain title
      content.getButtonText();//The action language (View details/downloads)
     content.getPrice();//Price (free/$6.3)
     content.getOther();//Other (2017-09-18 update)
}
```

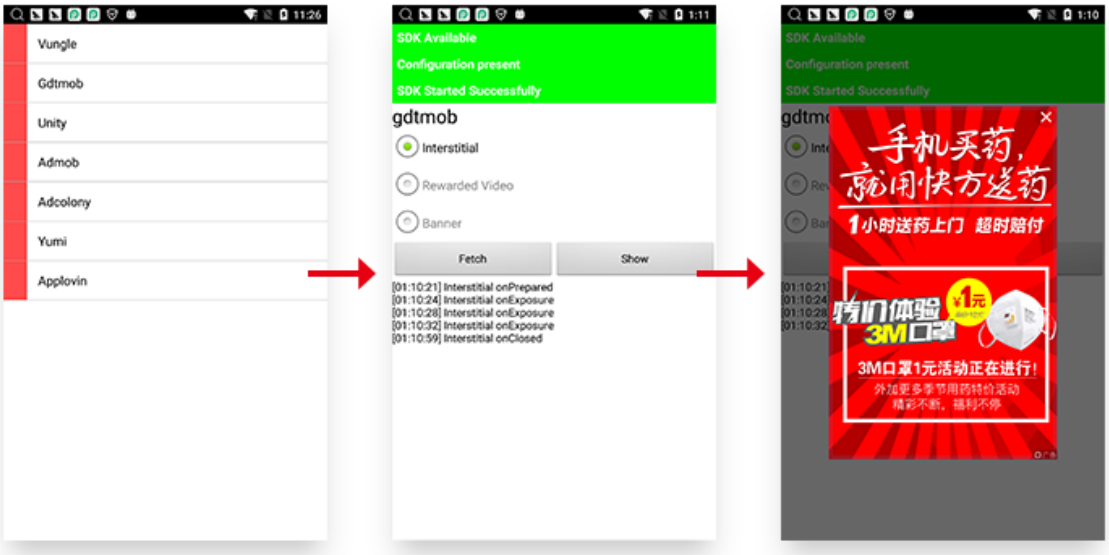
To ensure your revenue, please call the relevant reporting method at corresponding place.(Important)

```
content.reportShow(container,content); // report when impression happens (container refers to parent layout)

content.reportClick(container,content); // report when impression happens (container refers to parent layout)
```

4.6 Test Mode

YUMI SDK provideds a test mode to test your 3rd-party Integrations.



1. Call method to open test page :
   YumiSettings.startDebugging(Activity,BannerSlotID,InterstitialSlotID,MediaSlotID
);

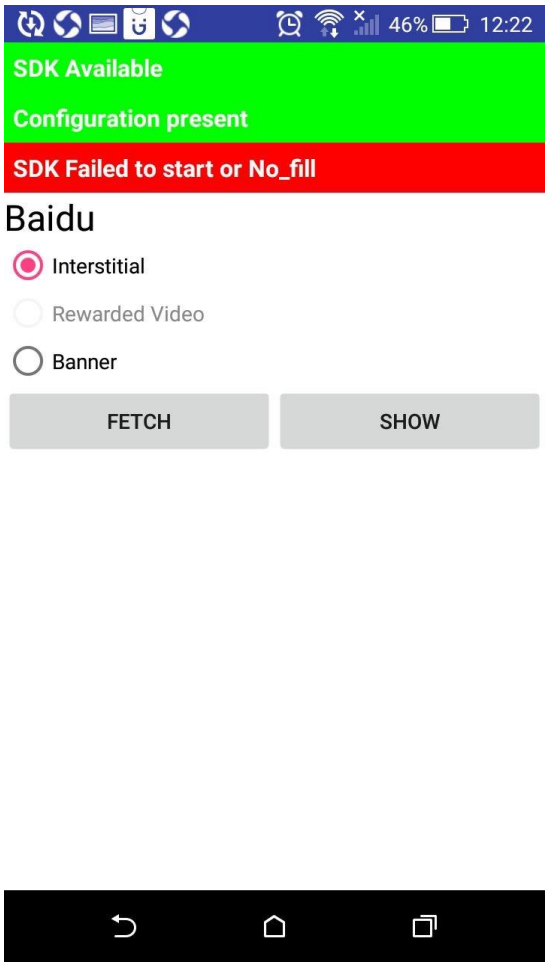
   Set channel according to your settings on the platform , You need to set up channelID、versionName :
   YumiSettings.startDebugging (Activity, BannerSlotID,InterstitialSlotID,MediaSlotID, channelID, versionName);
2. Debugging Home Screen – After basic account configuration on our website, and download of the SDK, all of the platforms should be listed on this first screen, initially in red. The red markers will turn green once a platform is properly integrated and (at least one ad type) is shown successfully within the debugging program. If none or only some are shown, please check with your operations rep for assistance.

Each platform line can be clicked to enter and investigate the various layers of testing available. The following items can be tested within the second screen for each platform shown on the debugging home screen.

a) Adapter SDK Present? When this first line is green, it means that the platform adapter has been added; when in red, it means that the adapter has not yet been added. If red, please review section 3.1-2 and check for or add any missing adapter.

b) Adapter Configuration Correct? When green, it means the platform adapter has been properly registered in the Manifest; when in red, it means that the adapter has not been added, is incomplete, or contains errors. Please see section 3.1.3, Exhibit B (if needed) along with any custom integration content generated on our website when the account was first configured. The platform adapter component content is essential.

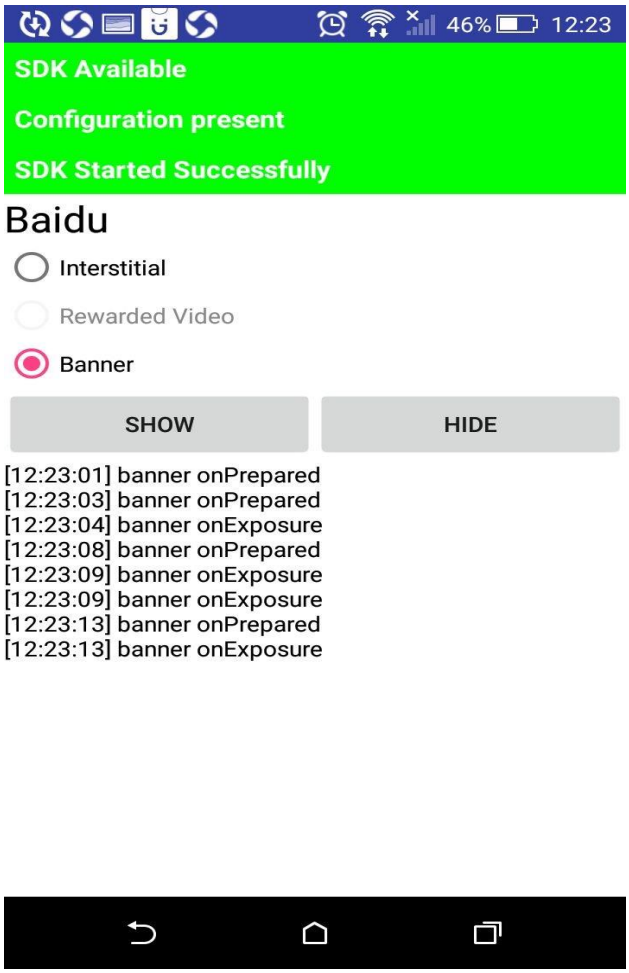
c) Ad Calls Initialized? You need to fetch an ad to test this line. When green, it means the ad platform was reached and an ad has been shown successfully; when in red, it means that the ad call was not successfully initialized or that there was no demand available from the platform for that request. Please email us if assistance required here.



d) Demand Available? Click Fetch to request ad. A fetch will generate an ad download. You will get a text response confirming the ad download, or instead an error.

e) Ad Displayed? Click Show to display an ad. When ad shows properly, all test elements will turn green, which indicates this platform has been integrated successfully, at least for that ad type.

Below is a sample screen of some banner ads that were fetched from Baidu ad network and displayed. The HIDE button simple allows the developer to test the SDK's hide feature.



f) The debugging mode must be completed before releasing the App.

4.7 Permissions for Android 6.0 and newer versions

When the targetSdkVersion of your app is 23 or above, you can choose the following method to check permission and prompt for user authorization.

Note: The default setting for this method is false, it won't prompt for user authorisation or causing crash. If set as true, it will check permission and prompt for user authorisation with popups. This method should be called before the instantiated ads and android-support-v4.jar needs to be added.

```
YumiSettings.runInCheckPermission(true);
```

4.8 Googleplay Version

Note: If your APP is Googleplay version, please do the following setting

```
YumiSettings.setAppIsGooglePlayVersions(true);
```

4.9 Proguard

If you are using Proguard add the following to your Proguard config file:
-keepattributes Exceptions , InnerClasses , Signature , Deprecated , SourceFile , LineNumberTable , *Annotation* , Synthetic , EnclosingMethod
-keep class com.yumi.android.sdk.ads.** { *;}
-keep class com.yumi.android.sdk.ads.self.**{*;}
-keep class com.yumi.android.sdk.ads.selfmedia.**{*;}