



# Android SDK User Guide 3.9

**SMART**  
mediation

**SMART**  
ad

**SMART**  
dialog

 Banner

리워드 배너

 Link

리워드 링크

# 애드립 앱 관리 및 mediation 개요

실제 프로젝트환경에 SDK 적용을 위한 문서입니다.

SDK 다운부터 API 키 발급까지에 대한 과정은 아래 링크에서 확인하실 수 있습니다.

<http://adlibr.com/howto/prepare>

애드립을 통해 실제 사용할 플랫폼은 프로젝트에서 선택적으로 포함하여 최종 바이너리 크기를 줄일 수 있습니다.

실제 테스트 프로젝트를 컴파일 하기 위하여

각 플랫폼 사이트에서 발급받은 APP - ID 및 각 OS 에 맞는 최신 SDK가 별도로 필요합니다.

기본적으로 테스트 프로젝트는 각 플랫폼의 jar 파일만 새로 링크하면 동작하도록 제작되었으며  
실제 플랫폼의 SDK 작동방식이 변하지 않는 한 `test.adlib.project.ads` 안의 내용을 수정할 필요는 없습니다.  
(실제 발급받은 광고 플랫폼 ID 적용 부분은 수정해야 합니다.)

# 컴파일을 위한 준비 #1

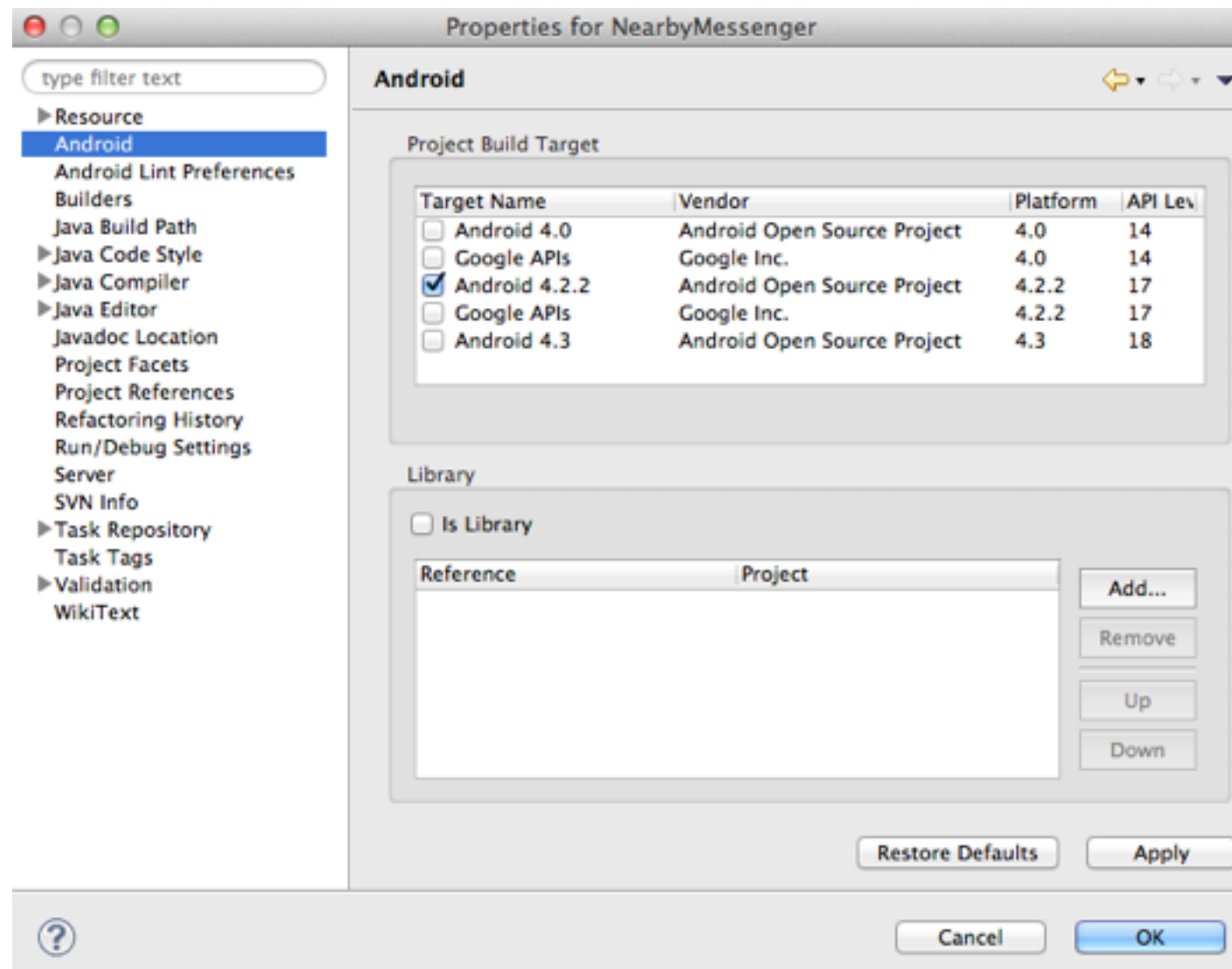
각 플랫폼의 가입과 SDK 다운로드를 개별적으로 진행합니다.

adlibrTestProject / libs 폴더를 열어 각 플랫폼의 jar 파일을 아래와 같이 모두 복사해 넣습니다.



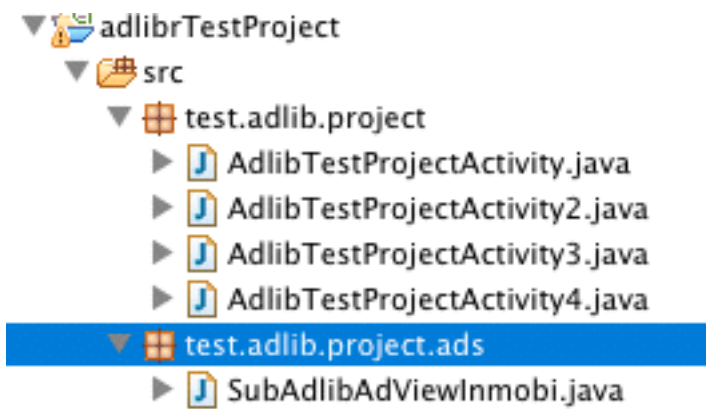
# 컴파일을 위한 준비 #2

프로젝트를 열어 아래와 같이 속성을 확인합니다.



Android 탭에서 사용할 Android SDK 버전을 체크합니다.  
애드립SDK 컴파일을 위해 API Level 17 이상의 Android SDK를 필요로 합니다.

# 프로젝트 자세히



기타 일반 플랫폼  
구현부

test.adlib.project 패키지는 테스트를 위한 Activity 가 위치하며  
test.adlib.project.ads 패키지는 각 광고 플랫폼의 실제 구현부 입니다.

test.adlib.project.ads 안에서 jar 파일과 마찬가지로  
실제 사용할 클래스만 선택하여 최종 파일 크기를 줄일 수 있습니다.

# AndroidManifest.xml

```
<uses-sdk android:minSdkVersion="8" />

<!-- 애드립 실행에 필요한 권한 각 플랫폼 별로 요청하는 권한이 모두 다르기 때문에 아래 권한들을 모두 추가하는것을 추천합니다. -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.GET_TASKS" />      <!-- 액티비티별 스케줄링을 위해 꼭 추가해주세요. -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<!-- 여기까지 애드립 사용을 위한 필수 권한 -->

<!-- 위치정보를 활용한 보다 최적화된 광고 송출을 위해 필요한 권한입니다. -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<!-- 위치정보를 활용한 보다 최적화된 광고 송출을 위해 필요한 권한입니다. -->

<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<!-- 애드립 실행에 필요한 권한 -->

<!-- 애드립 사용을 위해 꼭 추가해주세요. -->
<activity
    android:name="com.mocoplex.adlib.AdlibDialogActivity"
    android:theme="@android:style/Theme.Translucent"
    android:configChanges="orientation|keyboard|keyboardHidden" />

<activity android:name="com.mocoplex.adlib.AdlibWebBrowserActivity"
    android:configChanges="orientation|keyboard|keyboardHidden" />

<activity android:name="com.mocoplex.adlib.AdlibVideoPlayer"
    android:theme="@android:style/Theme.NoTitleBar"
    android:configChanges="orientation|keyboard|keyboardHidden" />
<!-- 애드립 사용을 위해 꼭 추가해주세요. -->
```

애드립의 기능 사용을 위해서  
반드시 추가 되어야 합니다.

그밖에 실제 프로젝트에 연동을 위하여 기본적으로 어플리케이션 구동에 필요한 권한 외에 위의 권한을 별도로 추가해주세요.

광고 플랫폼별 필요한 권한들은 각 플랫폼의 SDK 문서를 참조해주세요.

# 실제 프로젝트 연동 #1

보다 자세한 내용은 테스트 프로젝트의 `AdlibTestProjectActivity.java` 파일을 참조해주세요.

Adlib 을 연동하기 위해서 실제 Activity 구현부를 아래와 같이 수정합니다.

```
public class AdlibTestProjectActivity extends AdlibActivity
```

처음으로 실행되는 Activity 에서 광고 스케줄링 정보를 받아올 수 있도록 아래와 같이 먼저 호출합니다.

```
protected void initAds()  
{  
    // 광고 스케줄링 설정을 위해 아래 내용을 프로그램 실행시 한번만 실행합니다. (처음 실행되는 activity에서 한번만 호출해주세요.)  
    // 광고 subview 의 패키지 경로를 설정합니다. (실제로 작성된 패키지 경로로 수정해주세요.)  
    // 일반 Activity 에서는 AdlibManager 를 동적으로 생성한 후 아래 코드가 실행되어야 합니다. (AdlibTestProjectActivity4.java)  
  
    // 쓰지 않을 광고플랫폼은 삭제해주세요.  
    AdlibConfig.getInstance().bindPlatform("ADAM", "test.adlib.project.ads.SubAdlibAdViewAdam");  
    AdlibConfig.getInstance().bindPlatform("ADMOB", "test.adlib.project.ads.SubAdlibAdViewAdmob");  
    AdlibConfig.getInstance().bindPlatform("CAULY", "test.adlib.project.ads.SubAdlibAdViewCauly");  
    AdlibConfig.getInstance().bindPlatform("TAD", "test.adlib.project.ads.SubAdlibAdViewTAD");  
    AdlibConfig.getInstance().bindPlatform("NAVER", "test.adlib.project.ads.SubAdlibAdViewNaverAdPost");  
    AdlibConfig.getInstance().bindPlatform("SHALLWEAD", "test.adlib.project.ads.SubAdlibAdViewShallWeAd");  
    AdlibConfig.getInstance().bindPlatform("INMOBI", "test.adlib.project.ads.SubAdlibAdViewInmobi");  
    AdlibConfig.getInstance().bindPlatform("MMEDIA", "test.adlib.project.ads.SubAdlibAdViewMMedia");  
    AdlibConfig.getInstance().bindPlatform("MOBCLIX", "test.adlib.project.ads.SubAdlibAdViewMobclix");  
    AdlibConfig.getInstance().bindPlatform("ADMOBECPM", "test.adlib.project.ads.SubAdlibAdViewAdmobECPM");  
    AdlibConfig.getInstance().bindPlatform("UPLUSAD", "test.adlib.project.ads.SubAdlibAdViewUPlusAD");  
    AdlibConfig.getInstance().bindPlatform("MEZZO", "test.adlib.project.ads.SubAdlibAdViewMezzo");  
    AdlibConfig.getInstance().bindPlatform("AMAZON", "test.adlib.project.ads.SubAdlibAdViewAmazon");  
    AdlibConfig.getInstance().bindPlatform("ADHUB", "test.adlib.project.ads.SubAdlibAdViewAdHub");  
    // 쓰지 않을 플랫폼은 JAR 파일 및 test.adlib.project.ads 경로에서 삭제하면 최종 바이너리 크기를 줄일 수 있습니다.  
  
    // SMART* dialog 노출 시점 선택시 / setAdlibKey 키가 호출되는 activity 가 시작 activity 이며 해당 activity가 종료되면 app 종료로 인식합니다.  
    // adlibr.com 에서 발급받은 api 키를 입력합니다.  
    // https://sec.adlibr.com/admin/dashboard.jsp  
    AdlibConfig.getInstance().setAdlibKey("ADLIB API KEY");  
}
```

# 실제 프로젝트 연동 #1 자세히

1. 실제 구현된 광고 플랫폼 뷰를 프로젝트에 연결합니다.

```
AdlibConfig.getInstance().bindPlatform("INMOBI", "test.adlib.project.ads.SubAdlibAdViewInmobi");
```

위 패키지 경로는 실제 구현된 경로로 수정이 되어야하며 테스트 프로젝트에서는 `test.adlib.project.ads` 경로 아래에 구현되어 위와같이 연결되었습니다.

2. Adlib 에서 발급받은 API KEY 를 설정합니다.

```
AdlibConfig.getInstance().setAdlibKey("ADLIB API KEY");
```

API Key 는 <https://sec.adlibr.com/admin/dashboard.jsp> 에서 발급 및 확인이 가능합니다.  
노출, 클릭 관련 로그는 홈페이지 관리 페이지에서 실시간으로 확인 가능합니다.

3. 애드립을 이용하여 클라이언트 버전관리를 할 수 있습니다.

```
setVersionCheckingListner(new AdlibVersionCheckingListener());
```

서버에서 클라이언트 버전을 관리하여 사용자들에게 클라이언트 버전업을 유도할 수 있습니다.  
(자세한 내용은 샘플 프로젝트를 참조하세요.)  
클라이언트 최신 버전은 홈페이지 -> 설정 메뉴를 통해 설정 가능합니다.

deprecated : SMART\* dialog 를 통해 보다 손쉽게 앱 업데이트를 유도할 수 있습니다.



# 실제 프로젝트 연동 #2

애드립을 연동하기 위해 아래와 같이 구현합니다.

```
setContentView(R.layout.main);

initAds();

// 실제 광고 호출이 될 adview 를 연결합니다.
this.setAdsContainer(R.id.ads);
```

layout / main.xml 은 아래와 같이 구현되었습니다.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <!-- adlib adview -->
    <com.mocoplex.adlib.AdlibAdViewContainer
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/ads"
        isDefaultBanner="true" />

</LinearLayout>
```

isDefaultBanner : 초기 광고 로딩 전에 기본 하우스배너를 보여줄지 말지 여부.  
true 이면 애드립 배너 혹은 대쉬보드에서 설정하신 사용자배너가 초기 로딩 전에 보여집니다. (기본값은 true)

위와같이 xml 에서 `com.mocoplex.adlib.AdlibAdViewContainer` 로 구현된 id / ads 를

```
this.setAdsContainer(R.id.ads);
```

최종적으로 Activity 에 연결합니다.

# 실제 프로젝트 연동 #두번째 Activity

보다 자세한 내용은 테스트 프로젝트의 `AdlibTestProjectActivity2.java` 파일을 참조해주세요.

```
public class AdlibTestProjectActivity2 extends AdlibActivity {  
    /** Called when the activity is first created. */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main2);  
  
        this.setAdsContainer(R.id.ads);  
    }  
}
```

이미 전역적인 설정을 첫번째 Activity 에서 수행하였기 때문에 두번째 이후의 Activity 에서는 위와 같이 광고 뷰를 bind 하기만 하면 됩니다.

# 실제 프로젝트 연동 #광고 View 동적 생성

보다 자세한 내용은 테스트 프로젝트의 `AdlibTestProjectActivity3.java` 파일을 참조해주세요.

```
public class AdlibTestProjectActivity3 extends AdlibActivity
```

광고 뷰를 xml 이 아닌 아래와 같이 동적으로 코드상에서 생성 및 연결이 가능합니다.

```
// 동적으로 adview 를 생성합니다.  
avc = new com.mocoplex.adlib.AdlibAdViewContainer(AdlibTestProjectActivity3.this);  
  
ViewGroup vg = (ViewGroup)findViewById(R.id.maincontainer);  
vg.addView(avc);  
  
// 동적으로 생성한 adview 에 스케줄러를 바인드합니다.  
bindAdsContainer(avc);
```

초기 광고 로딩전에 하우스배너를 보여줄지 말지에 대한 설정을 하고 싶으시면,

```
avc = new com.mocoplex.adlib.AdlibAdViewContainer(AdlibTestProjectActivity3.this, true);
```

위와 같이 `AdlibAdViewContainer`를 생성할 때, boolean 값을 지정하면 됩니다.

true 이면 애드립 배너 혹은 대쉬보드에서 설정하신 사용자배너가 초기 로딩 전에 보여집니다.

boolean을 지정하지 않으면 기본으로 하우스배너가 노출됩니다.

# 실제 프로젝트 연동 #3

실제 광고 플랫폼의 뷰를 포함하는 패키지이며 `SubAdlibAdViewCore` 클래스를 상속받아 제작됩니다.

```
public void query()
```

스케줄러에 의해 위의 함수가 호출되며 광고를 수신한 경우

```
public void gotAd()
```

가 호출되며 위의 함수 호출로 실제로 `AdlibAdViewContainer` 를 통해 화면에 보여지게 됩니다.

위의 패키지명은 필요에 의해 임의의 프로젝트 경로로 변환이 가능하며  
변환된 경로는

```
AdlibConfig.getInstance().bindPlatform("INMOBI", "test.adlib.project.ads.SubAdlibAdViewInmobi");
```

를 통해 실제 경로로 변경해야합니다.

# 실제 프로젝트 연동 #4

테스트 프로젝트의 `test.adlib.project.ads` 패키지 안에는 제휴 및 일반 플랫폼의 view 가 구현되어 있습니다.

기본적인 모든 구현은 이미 완성되어 있으며 실제 각 플랫폼 구현부에서의 ID 부분만 실제 발급 받은 ID로 교체하여 사용합니다.

AndroidManifest.xml 에 ID 를 추가해야 하는 경우도 있습니다. (각 플랫폼의 SDK 문서 참조)

## ※주의※

- 1) Proguard를 이용하여 암호화 하는 경우 proguard configuration 파일 수정이 필요합니다.  
자세한 구현내용은 테스트 프로젝트의 proguard.cfg 파일

또는

<https://github.com/mocoplex/adlibr-SDK-android/blob/master/adlibrTestProject/proguard.cfg>  
위 주소를 참고해 주세요.

- 2) 애드립은 내부 스케줄링 정보를 저장하기 위해 document 폴더 안에 캐쉬 데이터를 저장합니다.

DocumentPath/Adlib-data/ 경로에 캐쉬 데이터를 저장하므로 해당 폴더를 삭제하지 말아주세요.

# AdlibActivity

```
public class AdlibActivity extends Activity {
    private AdlibManager _amanager;
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        _amanager = new AdlibManager();
        _amanager.onCreate(this);
    }
    protected void onResume()
    {
        _amanager.onResume(this);
        super.onResume();
    }
    protected void onPause()
    {
        _amanager.onPause();
        super.onPause();
    }
    protected void onDestroy()
    {
        _amanager.onDestroy(this);
        super.onDestroy();
    }
    public void setAdsContainer(int rid)
    {
        _amanager.setAdsContainer(rid);
    }
    public void bindAdsContainer(AdlibAdViewContainer a)
    {
        _amanager.bindAdsContainer(a);
    }
    public void loadInterstitialAd()
    {
        _amanager.loadInterstitialAd(this);
    }
    public void setVersionCheckingListner(AdlibVersionCheckingListener l)
    {
        _amanager.setVersionCheckingListner(l);
    }
    public void destroyAdsContainer()
    {
        _amanager.destroyAdsContainer();
    }
}
```

애드립은 기본적으로 Activity 를 상속받아  
각 status 별 최적화를 위한 별도 작업을 처리하게됩니다.  
기본 Activity 에서의 애드립 연동은 AdlibActivity 소스를 참고하여 구현 가능합니다.

AdlibTestProjectActivity4.java 파일을 참조해주세요.

# 애드립 팝 배너



- 그림과 같은 형태의 배너가 노출된 후 일정 시간이 지나면 사라집니다.

- 아래의 메소드를 호출하여 애드립 팝 배너를 보여줄 수 있습니다.

(AdlibActivity)

```
// 프레임 컬러 설정
setAdlibPopFrameColor(Color.DKGRAY);
// 버튼 컬러 설정 (BTN_WHITE, BTN_BLACK 2종류)
setAdlibPopCloseButtonStyle(AdlibPop.BTN_WHITE);
// in, out 애니메이션 설정 (ANIMATION_SLIDE, ANIMATION_NONE 2종류)
setAdlibPopAnimationType(AdlibPop.ANIMATION_SLIDE, AdlibPop.ANIMATION_SLIDE);
// 애드립팝 보이기 (int align, int padding(dp값))
// 정렬은 ALIGN_LEFT, ALIGN_TOP, ALIGN_RIGHT, ALIGN_BOTTOM 4종류
showAdlibPop(AdlibPop.ALIGN_BOTTOM, 50);
```

(AdlibManager 생성 시)

```
_amanager.setAdlibPopFrameColor(Color.DKGRAY);
_amanager.setAdlibPopCloseButtonStyle(AdlibPop.BTN_WHITE);
_amanager.setAdlibPopAnimationType(AdlibPop.ANIMATION_SLIDE, AdlibPop.ANIMATION_SLIDE);
_amanager.showAdlibPop(this, AdlibPop.ALIGN_BOTTOM, 50);
```

- 아래의 메소드를 호출하여 애드립 팝 배너를 숨길 수 있습니다.

(AdlibActivity)

```
hideAdlibPop();
```

(AdlibManager 생성 시)

```
_amanager.hideAdlibPop();
```

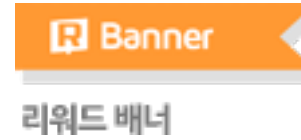
# 애드립 팝 배너

- 팝배너 수신 성공, 실패 등에 대한 이벤트 처리가 필요한 경우, handler 를 이용할 수 있습니다.

```
showAdlibPop(AdlibPop.ALIGN_BOTTOM, 30, new Handler() {  
    public void handleMessage(Message message) {  
        try  
        {  
            switch (message.what) {  
                case AdlibManager.DID_SUCCEED:  
                    Log.d("ADLIBr", "onReceivedPopBanner");  
                    break;  
                case AdlibManager.DID_ERROR:  
                    Log.d("ADLIBr", "onFailedPopBanner");  
                    break;  
                case AdlibManager.POP_CLOSED:  
                    Log.d("ADLIBr", "onClosedPopBanner");  
                    break;  
            }  
        }  
        catch(Exception e)  
        {  
        }  
    }  
});
```



# 애드립 리워드 배너 (띠배너)



- Background request 를 통해 지면낭비 없이, 더 높은 단가의 광고를 우선적으로 노출합니다.
- 별도 플랫폼의 회원가입이나 SDK 연동 없이, 바로 제휴 플랫폼의 광고를 효율적으로 노출할 수 있습니다.
- 애드립 리워드 배너는 실시간 단가 정보를 활용하여, 기존보다 높은 단가의 광고를 우선 노출하기 때문에 기존보다 더 높은 광고수익을 기대할 수 있습니다.

## 애드립테스트

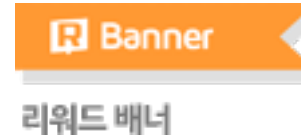


설정 후 저장 및 적용하기 버튼을 눌러야 반영됩니다.

저장 및 적용하기

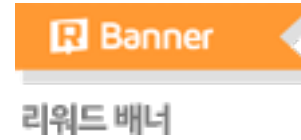
- 대쉬보드의 전역설정에서 R Banner 단가를 Auto 혹은 원하는 단가로 설정 후 “ON” 선택하여 “저장 및 적용하기” 버튼을 클릭합니다.
- 위 설정을 키는 것 만으로 바로 리워드배너를 앱에 노출할 수 있습니다.

# 애드립 리워드 배너(전면배너)



- 대쉬보드의 간단한 설정으로 “앱 시작시” 또는 “앱 종료시”에 전면배너를 자동 노출 할 수 있습니다.
- 그 외의 시점이나, 버튼 클릭 등과 같은 이벤트 발생 시에는 직접 호출하여 전면배너를 노출 할 수 있습니다.
- 직접 호출하여 전면배너를 노출할 때는 다양한 플랫폼을 스케줄링 할 수 있습니다.

# 애드립 리워드 배너(전면배너)



## 1. 자동노출

### 전면배너 간편 노출 설정

- 리워드배너 슬롯을 통해, 리포트 및 노출 스케줄설정 등, 다양한 기능이 지원되는 "사용자 리워드 전면배너" 기능을 이용할 수 있습니다.
- 별도의 코드상의 추가 구현없이 설정한 시점에 자동으로 전면배너가 노출됩니다. (SDK 3.5 이상)
- 트래픽 및 수익 내역은 D+1 "리워드 배너" 수익 리포트에서 확인할 수 있습니다.



앱 시작시

ON



앱 종료시

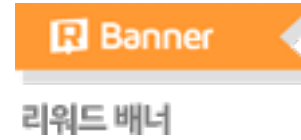
OFF

전면배너 스케줄링을 사용하려면, 여기를 클릭해주세요.

테스트 전면배너를 노출하려면, 여기를 클릭해주세요.

- 대쉬보드에서 “앱 시작시” 나 “앱 종료시” 를 ON으로 설정하시면, 아무런 구현 없이 해당 시점에 전면배너가 노출 됩니다.
- 단, 자동 노출 기능은 스케줄링을 사용할 수 없으며, 애드립에서 제공하는 전면배너 노출만 가능합니다.

# 애드립 리워드 배너(전면배너)



## 2. 직접 호출

전면배너 자동노출을 사용하지 않고 직접 호출을 하기 위해서는 아래 코드를 입력해주세요.

(AdlibActivity)

```
loadInterstitialAd();      - 다이얼로그 형태  
loadFullInterstitialAd(); - full size 전면배너
```

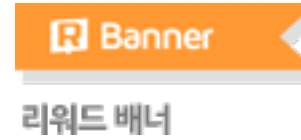
(AdlibManager 생성 시)

```
_amanager.loadInterstitialAd(this); - 다이얼로그 형태  
_amanager.loadFullInterstitialAd(this); - full size 전면배너
```

전면배너 수신 성공, 실패 등에 대한 이벤트 처리가 필요한 경우, handler 를 이용할 수 있습니다.

```
loadInterstitialAd(new Handler() {  
    public void handleMessage(Message message) {  
        try  
        {  
            switch (message.what) {  
            case AdlibManager.DID_SUCCEED:  
                Log.d("ADLIBr", "onReceiveAd " + (String)message.obj);  
                break;  
            case AdlibManager.INTERSTITIAL_FAILED:  
                Log.d("ADLIBr", "onFailedToReceiveAd");  
                break;  
            case AdlibManager.INTERSTITIAL_CLOSED:  
                Log.d("ADLIBr", "onClosedAd " + (String)message.obj);  
                break;  
            }  
        }  
        catch(Exception e)  
        {  
        }  
    }  
});
```

# 애드립 리워드 배너(전면배너)



## 2. 직접 호출 (스케줄링)

### 전면배너 간편 노출 설정

- 리워드배너 슬롯을 통해, 리포트 및 노출 스케줄설정 등, 다양한 기능이 지원되는 "사용자 리워드 전면배너" 기능을 이용할 수 있습니다.
- 별도의 코드상의 추가 구현없이 설정한 시점에 자동으로 전면배너가 노출됩니다. (SDK 3.5 이상)
- 트래픽 및 수익 내역은 D+1 "리워드 배너" 수익 리포트에서 확인할 수 있습니다.

앱 시작시

OFF

앱 종료시

OFF

전면배너 스케줄링 사용중

테스트 전면배너를 노출하려면, 여기를 클릭해주세요.

### 전면배너 노출 스케줄 사용자 설정

- 각 플랫폼의 native SDK 를 이용하여 애드립의 전면배너가 없을 경우 선택된 플랫폼의 전면배너가 노출됩니다.
- 플랫폼의 광고가 없는 경우 자동으로 다음 순서의 플랫폼이 노출됩니다.
- loadInterstitialAd() 함수를 호출한 시점에 전면배너 스케줄링이 동작합니다. (시작시/종료시 옵션은 동작안함)

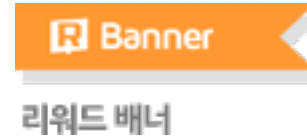
Cauly

Admob

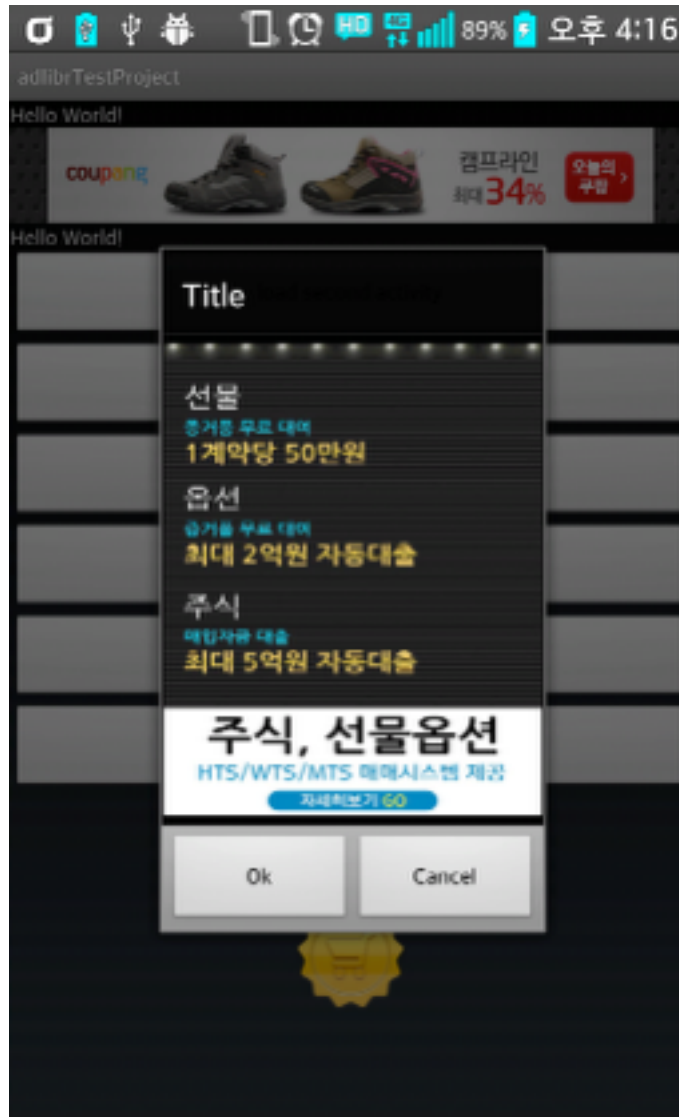
Adam

- 대쉬보드에서 전면배너 스케줄링 사용을 켜면, 스케줄 설정을 할 수 있습니다.
- 전면광고 수신에 실패하면 애드립전면배너 + 설정된 플랫폼 순서대로 전면광고를 호출합니다.

# 애드립 리워드 배너(전면배너)



## 3. 전면배너 view 활용

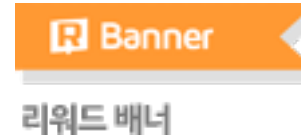


- 애드립 전면배너 View를 다른 형태로 이용할 수 있도록 API를 지원합니다.
- 아래의 메소드를 호출하여 AdlibInterstitialView를 원하는 곳에 사용할 수 있습니다.
- 자세한 내용은 샘플 프로젝트에서 확인할 수 있습니다.

```
// getInterstitialView(width(dp), maxHeight(dp), handler);
// width의 비율에 맞춰 height가 계산되어진 view를 반환합니다.
// 비율에 맞는 height가 maxHeight보다 클 경우에는 maxHeight에 맞춰 비율이 변형된 view를 반환합니다.
getInterstitialView(200, 300, new Handler() {
    public void handleMessage(Message message) {
        try
        {
            switch (message.what) {
                case AdlibManager.DID_SUCCEED:
                    AdlibInterstitialView iView = (AdlibInterstitialView)message.obj;

                    // getViewHeight 함수를 통해 실제 노출될 height 값(dp)을 알 수 있습니다.
                    int viewHeight = iView.getViewHeight();
                    break;
                case AdlibManager.DID_ERROR:
                    // 광고 수신 실패
                    break;
            }
        }
        catch(Exception e)
        {
        }
    }
});
```

# 애드립 리워드 배너 자세히



리워드 배너 노출을 위해 코드상에 추가로 구현할 내용은 없습니다.

AndroidManifest.xml 파일에 권한 및 activity 설정 내역이 추가되었는지 꼭 확인해주세요.

```
<!-- 애드립 실행에 필요한 권한 각 플랫폼 별로 요청하는 권한이 모두 다르기 때문에 아래 권한들을 모두 추가하는것을 추천합니다. -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.GET_TASKS" />      <!-- 액티비티별 스케줄링을 위해 꼭 추가해주세요. -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<!-- 여기까지 애드립 사용을 위한 필수 권한 -->

<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<!-- 애드립 실행에 필요한 권한 -->
```

```
<!-- 애드립 사용을 위해 꼭 추가해주세요. -->
<activity
    android:name="com.mocoplex.adlib.AdlibDialogActivity"
    android:theme="@android:style/Theme.Translucent"
    android:configChanges="orientation|keyboard|keyboardHidden" />

<activity android:name="com.mocoplex.adlib.AdlibWebBrowserActivity"
    android:configChanges="orientation|keyboard|keyboardHidden" />

<activity android:name="com.mocoplex.adlib.AdlibVideoPlayer"
    android:theme="@android:style/Theme.NoTitleBar"
    android:configChanges="orientation|keyboard|keyboardHidden" />
<!-- 애드립 사용을 위해 꼭 추가해주세요. -->
```



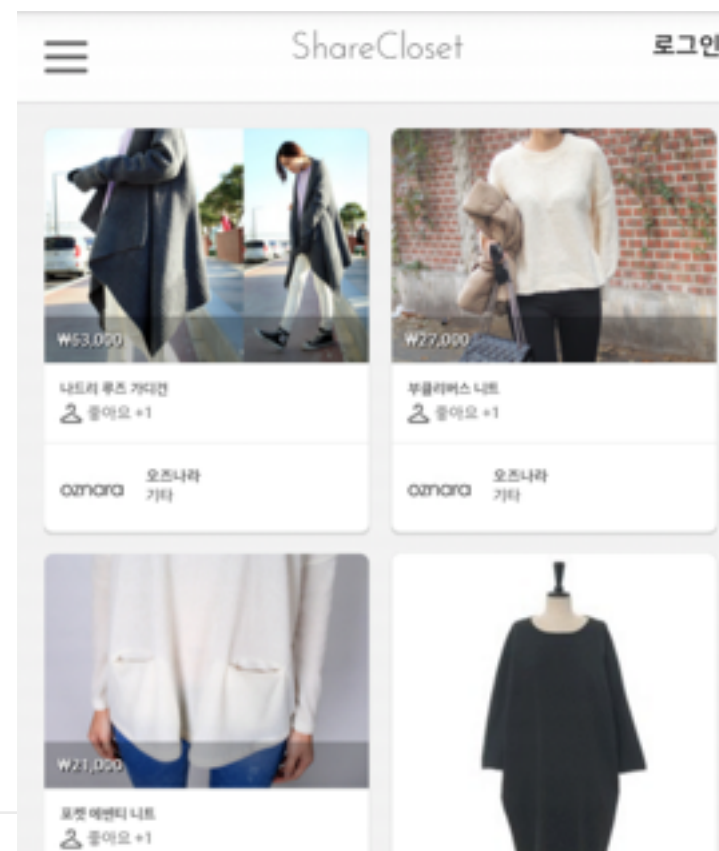
# 애드립 리워드 링크



## 애드립 리워드링크 개요



- 기존 Mediation 시스템과 별도로 동작하는 시스템입니다.
- 광고의 자유로운 위치 설정이 가능하며, 아이콘 형태로 최소한의 지면을 사용합니다.
- 배너광고 이외의 수익을 추가로 더 얻을 수 있습니다.
- 노출 여부를 관리자 페이지에서 손쉽게 설정할 수 있습니다.





# 애드립 리워드 링크



애드립 리워드 링크는 기존의 mediation 시스템과 개별적으로 동작합니다.




애드립 리워드 링크를 이용해, 앱에 [ 아이콘 ] 형태의 광고를 노출할 수 있으며,  
[ 아이콘 ] 광고를 이용하여 배너 수익외 추가 수익을 거둘 수 있습니다.

리워드 링크의 카테고리 상품은 계속적으로 확대 적용될 예정이며, 매뉴얼에서는 [ 리워드 패션 링크 ] 적용 샘플을 안내합니다.

애드립 리워드 링크를 앱에 적용하기 위해서는



메뉴를 통해서 리워드 링크 API 키를 발급받습니다.

링크 이름	<input type="text" value="test"/>
링크 ID	<input type="text" value="5184d07ae4b03c9009dfa5ae"/>
노출 아이콘	<div></div>
아이콘 Width (dp)	<input type="text" value="60"/>
아이콘 Height (dp)	<input type="text" value="60"/>
노출 여부	<input checked="" type="checkbox"/> ON

노출할 아이콘을 선택하고 노출 여부를 결정한 후  
실제 앱 구현부에 발급받은 API 키를 적용합니다.

# 리워드링크 사용법 - 구현

리워드링크 아이콘을 노출하기 원하는 Activity의 onResume과 onPause에 아래와 같이 호출합니다.

애드립 mediation 과 별도로 동작하며, 샘플 프로젝트를 참조하여 노출을 원하는 Activity 에 2라인만 추가하면 모든 적용이 완료됩니다.

```
@Override
public void onResume() {
    AdlibRewardLink.getInstance().rewardLink(this, "REWARD_LINK_KEY", x, y, align);
    super.onResume();
}
```

```
@Override
protected void onPause() {
    AdlibRewardLink.getInstance().pauseRewardLink(this);
    super.onPause();
}
```

x - x축 padding.  
pixel 단위의 int 양수값

y - y축 padding.  
pixel 단위의 int 양수값

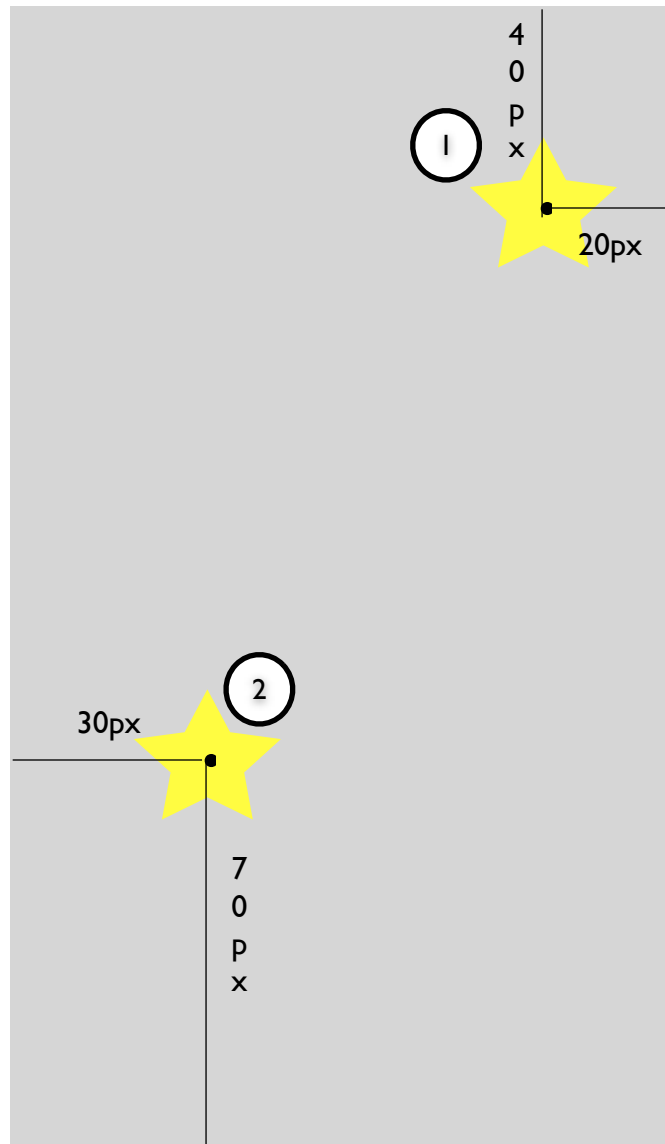
align - 아이콘 정렬 기준 .

AdlibRewardIcon.ALIGN_LEFT_TOP	좌측 상단
AdlibRewardIcon.ALIGN_RIGHT_TOP	우측 상단
AdlibRewardIcon.ALIGN_LEFT_BOTTOM	좌측 하단
AdlibRewardIcon.ALIGN_RIGHT_BOTTOM	우측 하단

아이콘 align의 기준값에서 x축으로 x만큼 이동, y축으로 y만큼 이동한 곳에 아이콘이 배치됩니다.

# 리워드링크 사용법 - 구현

리워드링크 아이콘 사용예제 (\*)아이콘 위치는 아이콘의 중심점 기준입니다.



- ① `AdlibRewardLink.getInstance().rewardLink(this, "KEY", 20, 40, AdlibRewardIcon.ALIGN_RIGHT_TOP);`
- ② `AdlibRewardLink.getInstance().rewardLink(this, "KEY", 30, 70, AdlibRewardIcon.ALIGN_LEFT_BOTTOM);`

# 리워드링크 사용법 - 구현2

- 용도에 따라 아이콘을 직접 add, remove 할 수 있도록 View를 return 하는 API를 지원합니다.
- 아래의 메소드를 호출하여 AdlibRewardIconView를 원하는 곳에 사용할 수 있습니다.
- 자세한 내용은 샘플 프로젝트에서 확인할 수 있습니다.

```
AdlibRewardLink.getInstance().getRewardLinkInfo(this, "REWARD_LINK_KEY", new Handler() {  
    public void handleMessage(Message message) {  
        try  
        {  
            switch (message.what) {  
                case AdlibManager.DID_SUCCEED:  
                    AdlibRewardIconView iconView = (AdlibRewardIconView)message.obj;  
  
                    // view의 width 값  
                    int w = iconView.getWidthSize();  
                    // view의 height 값  
                    int h = iconView.getHeightSize();  
  
                    break;  
                case AdlibManager.DID_ERROR:  
                    break;  
            }  
        }  
        catch(Exception e)  
        {  
        }  
    }  
});
```

# 리워드링크 사용법 - 직접 호출 예제

## 리워드링크 직접 호출 예제

아이콘 노출이 아닌 앱의 별도 버튼이나 메뉴를 통해 리워드링크의 랜딩 페이지를 호출할 수 있습니다.

버튼 혹은 메뉴 클릭시 호출되는 handler 에 아래 코드를 입력해주세요.

```
AdlibRewardLink.getInstance().showRewardLink(this, "REWARD_LINK_KEY");
```

자세한 내용은 샘플 프로젝트에서 확인하실 수 있습니다.

# 애드립 리워드 링크 자세히



리워드 링크 노출을 위한 코드 구현이 마무리 되었다면

AndroidManifest.xml 파일에 권한 및 activity 설정 내역이 추가되었는지 꼭 확인해주세요. (Manifest 설정은 mediation 과 공유합니다.)

```
<!-- 애드립 실행에 필요한 권한 각 플랫폼 별로 요청하는 권한이 모두 다르기 때문에 아래 권한들을 모두 추가하는것을 추천합니다. -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.GET_TASKS" />      <!-- 액티비티별 스케줄링을 위해 꼭 추가해주세요. -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<!-- 여기까지 애드립 사용을 위한 필수 권한 -->

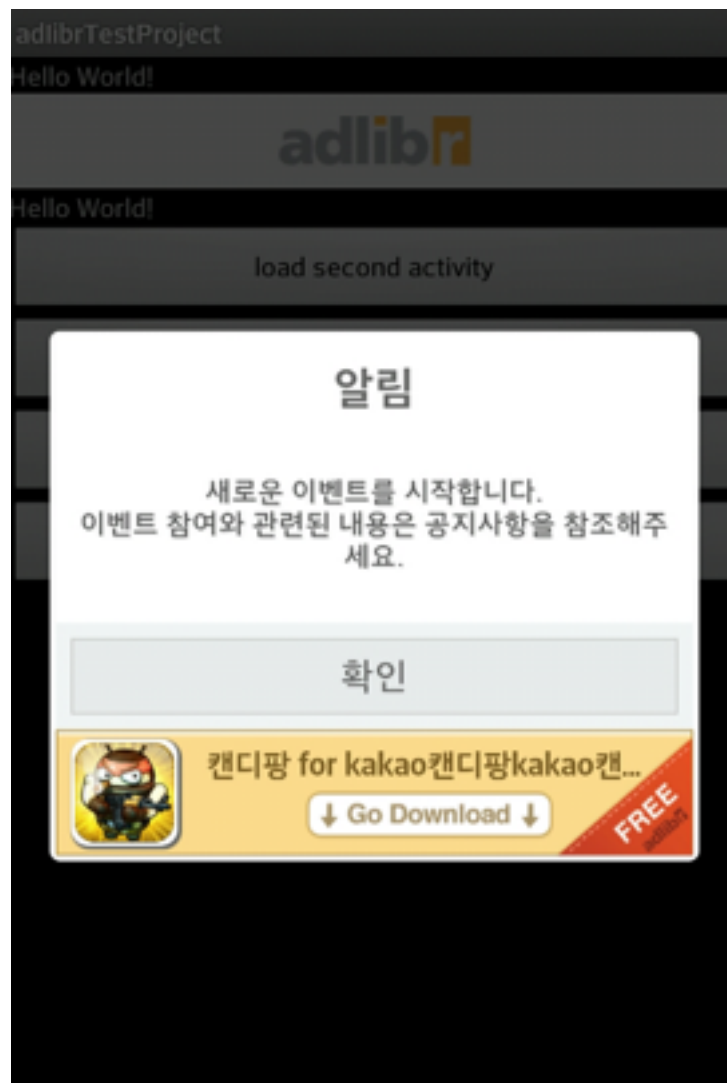
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<!-- 애드립 실행에 필요한 권한 -->
```

```
<!-- 애드립 사용을 위해 꼭 추가해주세요. -->
<activity
    android:name="com.mocoplex.adlib.AdlibDialogActivity"
    android:theme="@android:style/Theme.Translucent"
    android:configChanges="orientation|keyboard|keyboardHidden" />

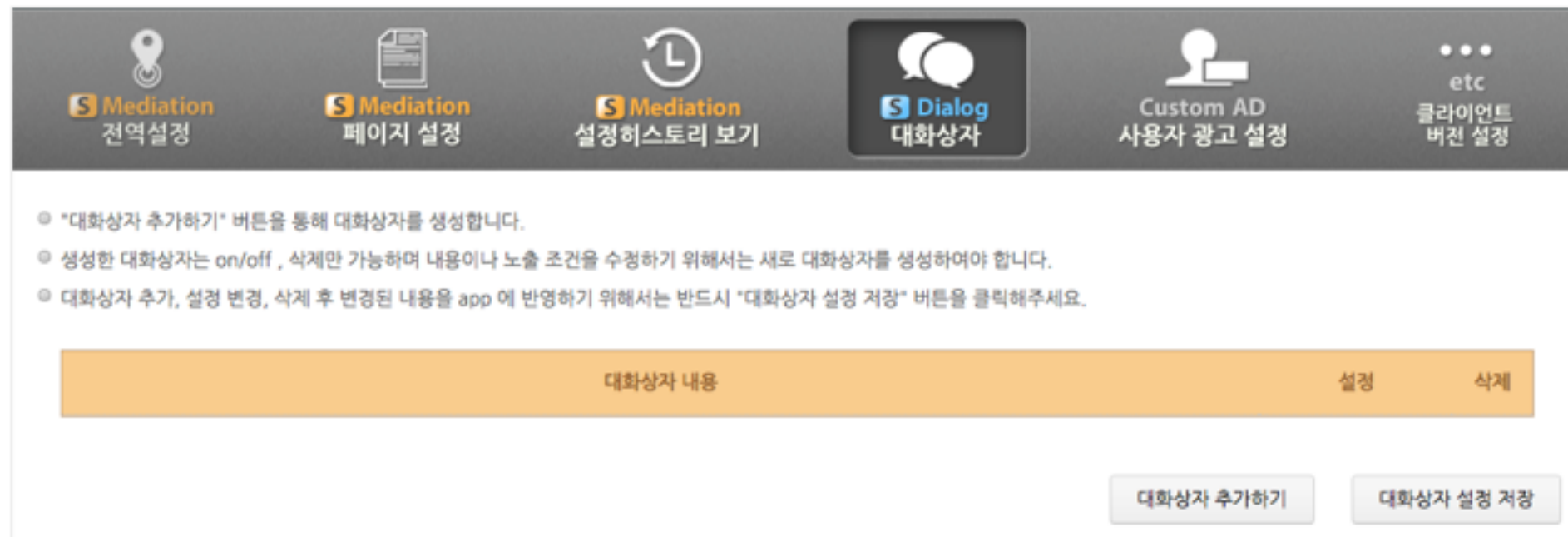
<activity android:name="com.mocoplex.adlib.AdlibWebBrowserActivity"
    android:configChanges="orientation|keyboard|keyboardHidden" />

<activity android:name="com.mocoplex.adlib.AdlibVideoPlayer"
    android:theme="@android:style/Theme.NoTitleBar"
    android:configChanges="orientation|keyboard|keyboardHidden" />
<!-- 애드립 사용을 위해 꼭 추가해주세요. -->
```

## Smart Dialog 란?



- 공지사항 알림, 앱 업데이트 유도 등 개인화된 대화상자를 별도 구현없이 손쉽게 생성 할 수 있습니다.
- 온라인에서의 간단한 설정만으로 손쉽게 대화상자를 컨트롤 할 수 있습니다.
- 대화상자를 통해 배너가 노출될 때 마다 3,000여개의 앱이 사용중인 애드립 네트워크 안에서 나의 앱도 교차 홍보되어, 더 많은 사용자들에게 나의 앱을 알릴 수 있습니다.



- 대화상자는 대쉬보드의 대화상자 탭에서 "대화상자 추가하기" 를 클릭 후 원하는 형태로 설정하여 노출할 수 있습니다.
- 노출시점 설정시, App 시작시점은 프로젝트에서의 소스 구현분의  
`AdlibConfig.getInstance().setAdlibKey("ADLIB API KEY");`  
 해당 소스가 구현되어 있는 Activity가 create 되는 시점이며, App 종료시점 역시 같은 Activity가 destroy 되는 시점입니다.



# 자주 묻는 질문 FAQ

## - 꼭 AdlibActivity 를 상속받아 구현해야 하나요?

애드립과 관련한 기본적인 구현은 **AdlibManager** 클래스에서 구현되었으며 이 클래스를 멤버로 Activity status 에 맞추어 멤버 함수를 호출하여 사용 가능합니다.  
<https://github.com/mocoplex/adlibr-SDK-android/blob/master/adlibrTestProject/src/test/adlib/project/AdlibTestProjectActivity4.java>  
위 주소에서 자세한 구현 내역을 확인 가능합니다.

## - 각 나라에 맞게 광고 스케줄을 달리 적용하고 싶습니다. (지역 타게팅)

애드립의 앱 스케줄 설정 화면에서, 타게팅을 원하는 나라를 추가하여 손쉽게 노출전략을 설정할 수 있습니다.

## - proguard 적용 후 광고, 대화상자가 보이지 않습니다.

실제 바인드하는 패키지의 난독화로 광고가 보이지 않는 현상이 나타날 수 있습니다.  
-keep 옵션을주어 아래코드에서 사용된 패키지 경로의 난독화를 방지합니다.

자세한 구현내용은 테스트 프로젝트의 proguard.cfg 파일

또는

<https://github.com/mocoplex/adlibr-SDK-android/blob/master/adlibrTestProject/proguard.cfg>  
위 주소를 참고해 주세요.

# 정리

애드립 연동이 마무리되었습니다.

## 실제 프로젝트에 애드립을 연동하기 위해 ##

1. 각 플랫폼의 ID 발급 및 SDK 다운로드
2. SDK 프로젝트에 라이브러리 추가
3. AndroidManifest.xml 수정
4. `test.adlib.project.ads` 안의 광고 ID 설정 부분 수정
5. 광고를 보여줄 Activity 의 `AdlibActivity` 상속  
(일반 Activity의 경우에는 `AdlibManager` 생성 및 `onCreate`, `onResume`, `onPause`, `onDestroy` 호출)
6. `AdlibConfig.getInstance()` 를 통한 전역적인 광고 설정  
(일반 Activity의 경우 이 작업보다 `AdlibManager`의 생성 및 `onCreate`가 먼저 이루어져야 합니다.)
7. `AdlibAdViewContainer` 생성 및 bind

관련된 추가적인 개발관련한 문의사항과 고급설정 등 관련 내용은  
<http://adlibr.com> 의 개발자 지원 메뉴를 통해 공유되고 있습니다.

감사합니다.