# StartApp In-App Ads Integration v2.1

## Introduction

This document will guide you through the integration process of the StartApp in-app ads, which will allow you to make money from your Android applications.
Once integrated, the SDK will allow you to enjoy StartApp's in-app monetization products, offering you the opportunity to maximize the revenue from your application. All this with minimal interference to the user experience.
**If you have any questions, contact us via support@startapp.com**

## SDK integration steps

## Step 1: Add the SDK JAR to your Eclipse project

Copy the SDK jar file from the SDK zip to the “libs” directory of your project.

## Step 2: Update your manifest file

Under the **main** manifest tag, add the following permissions:

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

//These permissions are only required for showing the ad when pressing the Home button:
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.GET_TASKS"/>
```

Under the **application** tag, add new activities:
Note: replace <**package_name**> with your package as declared in your manifest in both activities.

```
<activity android:name="com.startapp.android.publish.list3d.List3DActivity"
android:taskAffinity="<package_name>.AppWall"
android:theme="@android:style/Theme" />

<activity android:name="com.startapp.android.publish.AppWallActivity"
android:theme="@android:style/Theme.Translucent"
android:taskAffinity="<package_name>.AppWall"
android:configChanges="orientation|keyboardHidden" />
```

## Step 3: Initialize StartApp Ad

In the `OnCreate` method of your activity, call the static function:

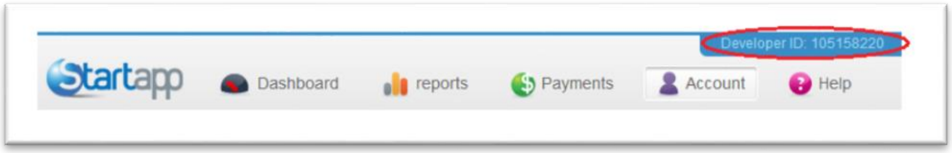
```
StartAppAd.init(this, "<Your Developer Id>", "<Your App ID>");
```

before calling setContentView()

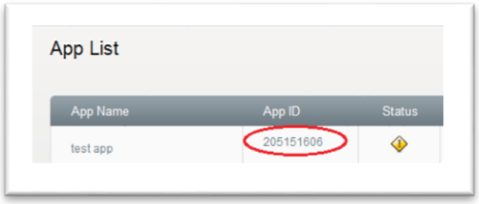
Note: The parameters of `StartAppAd.init` are:

1. Context – Activity context
2. Developer ID – String
3. App ID – String

You can find your IDs in the developers' portal: http://developers.startapp.com
After logging in, your developer ID will be at the top right-hand corner of the page:



To find your application ID, click on Dashboard and then choose the relevant ID from your app list:


App List

| App Name | App ID | Status |
|---|---|---|
| test app | 205151606 | |

## Step 4: Show Banners

There are 3 different types of banners:

| **Banner Type** | **Description** |
|---|---|
| Automatic Banner (recommended) | An automatic selection of banners between the two listed below |
| Standard Banner | A Standard Banner |
| 3D Banner | A three dimensional rotating banner |

**Adding the Automatic Banner**

To add the Automatic Banner, add the following view inside your Activity layout XML:

```
<com.startapp.android.publish.banner.Banner
        android:id="@+id/startAppBanner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"/>
```

Note: This code will place a View inside your Activity and you can add additional attributes for placing it in the desired location within the Activity.

**If you do not wish to add the Automatic Banner, choose one of the following options:**

1. **Adding a Standard Banner**
   Add the following View inside your Activity layout .XML

```
<com.startapp.android.publish.banner.bannerstandard.BannerStandard
    android:id="@+id/startAppStandardBanner"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"/>
```

   Note: This code will place a View inside your Activity and you can add additional attributes for placing it in the desired location within the Activity.

2. **Adding a 3D Banner**
   Add the following View inside your Activity layout .XML:

```
<com.startapp.android.publish.banner.banner3d.Banner3D
    android:id="@+id/startApp3DBanner"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"/>
```

   Note: This code will place a View inside your Activity and you can add additional attributes for placing it in the desired location within the Activity.

## Step 5: Show Interstitial Ads

**Initializing the StartApp Ad Object**

1. In your activity, create a member variable:

```
private StartAppAd startAppAd = new StartAppAd(this);
```

   Note: The parameter of `startAppAd` constructor is the context (activity).

2. Override the `onResume` method and add the call to `startAppAd.onResume()`:

```
@Override
public void onResume(){
    super.onResume();
    startAppAd.onResume();
}
```

   Note: Add this call right after the call to `super.onResume()`

**Showing Interstitials:**

1. **Show the Ad in chosen places within the app**
   You can choose to show the interstitial ad in several locations within your application.
   This could be upon entering, between stages, while waiting for an action and more.

   We do, however, recommend showing the ad upon exiting the application by using the 'back' button or the 'home' button, as explained in steps 2 and 3 below.

   Add the following code to the appropriate place or places within your activities in which you would like to show the ad:

```
startAppAd.showAd(); // show the ad
startAppAd.loadAd(); // load the next ad
```

   Note: Don't forget to call loadAd() right after showAd() – this will load your next ad.

   Example for showing an interstitial ad between activities:

```
public void btnOpenActivity (View view){
    startAppAd.showAd();
    startAppAd.loadAd();
    Intent nextActivity = new Intent(this, NextActivity.class);
    startActivity(nextActivity);
}
```

2. **Show the Ad upon exit by pressing the 'back' button**
   Override the `onBackPressed()` method and add a call to the `startAppAd.onBackPressed()`:

```
@Override
public void onBackPressed() {
    startAppAd.onBackPressed();
    super.onBackPressed();
}
```

   Note: Place the `startAppAd.onBackPressed()` call BEFORE the `super.onBackPressed()` call.

3. **Show the Ad upon exit by pressing 'home' button**
   The Home button functionality can improve results and revenue.
   Override the `onPause()` method and add a call the `startAppAd.onPause()`:

```
@Override
public void onPause() {
    super.onPause();
    startAppAd.onPause();
}
```

   Notes:

   a. There are two extra permissions required to run this as described in "Step 2: Update your manifest file" above.
   b. To display the ad in more activities, simply repeat these steps in each desired activity.

## Step 6: Show Splash Ad

The splash ad unit includes a full page splash screen followed by a full page ad.

There are two different modes of displaying a splash screen:

| **Splash Screen Mode** | **Description** |
|---|---|
| Template mode | A pre-defined template with your application name, logo, and loading animation. |
| User-Defined mode | Splash screen provided by the developer as a layout. |

**Adding the Template splash screen**

In the `OnCreate` method of your activity, after calling `StartAppAd.init` and before `setContentView`, call the static function:

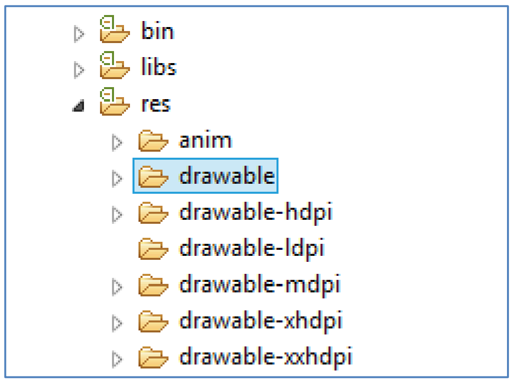
```
StartAppAd.showSplash(this, savedInstanceState,
            new SplashConfig()
                  .setTheme(SplashConfig.Theme.BLAZE)      // using the "BLAZE" theme
                  .setLogo(R.drawable.your_360x360_logo)   // resource ID
 );
```

- First parameter  is the context (activity)
- Second parameter `savedInstanceState` is the Bundle parameter passed to your `onCreate(Bundle savedInstanceState)` method.
- Third parameter is a "SplashConfig" object which can be used to customize some of your template's properties to suit your needs, such as your application name, logo and theme. For the full SplashConfig API please refer to Appendix B.

Example:
Custom template with OCEAN theme, modified application name, logo and landscape orientation:

```
StartAppAd.showSplash(this, savedInstanceState,
            new SplashConfig()
                   .setTheme(SplashConfig.Theme.OCEAN)
                   .setAppName("Yout Application Name")
                   .setLogo(R.drawable.your_360x360_logo)   // resource ID
                   .setOrientation(SplashConfig.Orientation.LANDSCAPE)
);
```

Note: For optimal appearance of your splash screen on all device's densities, please provide a logo of 360x360px and place it under your "*drawable*" folder in your project (in case this folder doesn't exists, you should create one).
In case you don't provide a logo, the SDK will use the default application icon as declared in the manifest and stretch it to 360x360px).



**Adding User-Defined splash screen**

If you already have a splash screen for your application or want to design a custom layout by yourself, you can do this by setting a SplashConfig object with a specific layout resource ID, and passing it to `showSplash` static function. For the full SplashConfig API please refer to Appendix B.

Example:

```
StartAppAd.showSplash(this, savedInstanceState,
            new SplashConfig()
                    .setTheme(SplashConfig.Theme.USER_DEFINED)
                    .setCustomScreen(R.layout.your_splash_screen_layout_id)
 );
```

## Step 7: Obfuscation (optional)

StartApp SDK is already obfuscated. If you choose to obfuscate your App by using proguard, you need to use the following configuration in the proguard configuration file:

```
-keep class com.startapp.** {
      *;
}
-keepattributes Exceptions, InnerClasses, Signature, Deprecated,  SourceFile,
 LineNumberTable, *Annotation*, EnclosingMethod
-dontwarn android.webkit.JavascriptInterface
-dontwarn com.startapp.**
```

# Appendixes

## Appendix A: Advanced Usage

### Adding Callback when Ad has loaded

`startAppAd.loadAd()` can get an implementation of `AdEventListener` as a parameter.
In case you want to get a callback for the ad load, pass the object which implements `AdEventListener` (this can be your activity) as a parameter to the method. This object should implement the following methods:

```
@Override
public void onReceiveAd(Ad ad) {
}

@Override
public void onFailedToReceiveAd(Ad ad) {
}
```

**Example:**

```
startAppAd.loadAd (new AdEventListener() {
    @Override
    public void onReceiveAd(Ad ad) {
    }

    @Override
    public void onFailedToReceiveAd(Ad ad) {
    }
});
```

### Adding Callback when Ad has been shown

`startAppAd.showAd()` can get an implementation of AdDisplayListener as a parameter.
In case you want to get a callback for the ad show, pass the object which implements AdDisplayListener (this can be your activity) as a parameter of the method. This object should implement the following methods:

```
@Override
public void adHidden(Ad ad) {
}

@Override
public void adDisplayed(Ad ad) {

}
```

**Example:**

```
startAppAd.showAd(new AdDisplayListener() {
    @Override
    public void adHidden(Ad ad) {
    }
    @Override
    public void adDisplayed(Ad ad) {
    }
});
```

**Explicitly selecting the type of Ad to load**

`startAppAd.loadAd()` can be told to decide which Ad to load for later use with the AdMode parameter, The options for this parameter are:

| Parameter Name | Description | Specific Ad Load Example |
|---|---|---|
| AUTOMATIC (recommended) | Auto selection of the best next interstitial to display | `startAppAd.loadAd(AdMode.AUTOMATIC)` |
| FULLPAGE | A full-page interstitial | `startAppAd.loadAd(AdMode.FULLPAGE)` |
| OFFERWALL | An automatic selection between a standard and a 3D offerwall. | `startAppAd.loadAd(AdMode.OFFERWALL)` |
| OVERLAY | An overlay interstitial | `startAppAd.loadAd(AdMode.OVERLAY)` |

The default value of this parameter is "AUTOMATIC" which will select the ad with the best performance.

When using this mode, additional methods in the activity life cycle must be implemented:

1. Override the `onSaveInstanceState(Bundle outState)` method and add a call to `startAppAd.onSaveInstanceState(outstate)`:

   Note: Add this call right after the call to `super.onSaveInstanceState(outState)`.

   Example:
```
@Override
protected void onSaveInstanceState (Bundle outState){
    super.onSaveInstanceState(outState);
    startAppAd.onSaveInstanceState(outState);
 }
```

2. Override the `onRestoreInstanceState(Bundle savedInstanceState)` method and add a call to `startAppAd.onRestoreInstanceState(savedInstanceState)`:

   Note: Add this call right BEFORE the call to `super.onRestoreInstanceState(savedInstanceState.`

   Example:
```
@Override
protected void onRestoreInstanceState (Bundle savedInstanceState){
    startAppAd.onRestoreInstanceState(savedInstanceState);
    super.onRestoreInstanceState(savedInstanceState);
}
```

**Explicitly Closing Interstitial Ad**

You can explicitly close the interstitial ad by calling:

`startAppAd.close();`

This will close the ad and return the control to the calling Activity. You can use this when implementing a timeout for an ad.

Note:  Keep in mind that the user can close the ad before timeout expires.

# Appendix B: SplashConfig API

### public SplashConfig setTheme(SplashConfig.Theme theme)

Set the splash theme – template mode or user-defined mode

**Parameters**

*SplashConfig.Theme.DEEP_BLUE (default)*
*SplashConfig.Theme.SKY*
*SplashConfig.Theme.ASHEN_SKY*
*SplashConfig.Theme.BLAZE*
*SplashConfig.Theme.GLOOMY*
*SplashConfig.Theme.OCEAN*
*SplashConfig.Theme.USER_DEFINED – user-defined mode*

### public SplashConfig setCustomScreen(**int** resource)

Set the splash layout for the Custom mode (mandatory if using the SplashConfig.Theme.USER_DEFINED)

**Parameters**

*Layout Resource ID*

### public SplashConfig setAppName(String appName)

Set the application name to be used in the template mode

**Parameters**

*String (default is the application name from the manifest).*

### public SplashConfig setLogo(int resource)

Set the logo to be displayed in the template mode

**Parameters**

*Drawable resource ID (default is the icon resource from the manifest)*

### public SplashConfig setOrientation(SplashConfig.Orientation orientation)

Set the orientation to be used in the template and user-defined modes

**Parameters**

*SplashConfig.Orientation.PORTRAIT (default)*
*SplashConfig.Orientation.LANDSCAPE*
*SplashConfig.Orientation.AUTO – use the device's orientation upon entering the application*