



## Giftiz SDK Integration Guide 1.4.9

### Android

#### Contents

1. Foreword.....	2
2. Giftiz SDK content.....	2
3. Step 1 / 4 – Link your project with Giftiz SDK Library.....	2
4. Step 2 / 4 – Update the AndroidManifest.xml of your App.....	3
a) Permissions .....	3
b) Application – Partner Key and GiftizWebViewActivity .....	3
c) Application – Logs.....	3
5. Step 3 / 4 – Call SDK Methods.....	3
a) onResumeMainActivity + onPauseMainActivity .....	3
b) missionComplete .....	4
c) InApp purchases .....	4
d) Paid Apps.....	4
6. Step 4 / 4 – Add the Giftiz Button on your main menu screen.....	4
7. Congratulations! .....	5
8. If you don't want to use the GiftizButtonView.....	6
a) Button Status.....	6
b) Button Clicked .....	6
c) Button status listener .....	6



## 1. Foreword

Giftiz is a standalone app that connects to partner android apps like yours. Users earn XP points on Giftiz when they achieve a specific mission in your app. They earn Giftiz [virtual currency] when they make a purchase in your app. XP points and Giftiz [virtual currency] are necessary to win real gifts on Giftiz [the app].

If you encounter any problem or if you like to give us feedback, feel free to contact us.

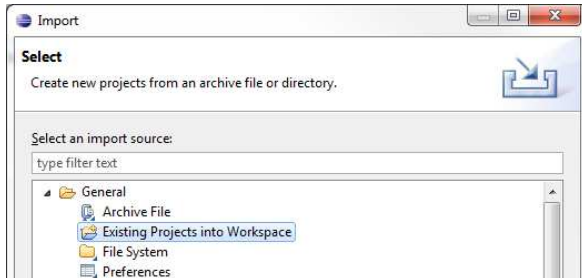
Your personal contact is: [etienne@giftiz.com](mailto:etienne@giftiz.com) (French & English)

## 2. Giftiz SDK content

GiftizSDK\_1.4.9.zip package contains 4 folders:

- **Customize:** All you need to customize the SDK Buttons.
- **GiftizSDKExamplePartner:** A very simple example of an Android App that integrates the SDK. Please refer to it if you have any doubts.
- **GiftizSDKExamplePartner\_SelfManaged:** Same example but with the Giftiz Button self-managed. See paragraph 8.
- **GiftizSDKLibrary:** The actual SDK, it's an Android Library.

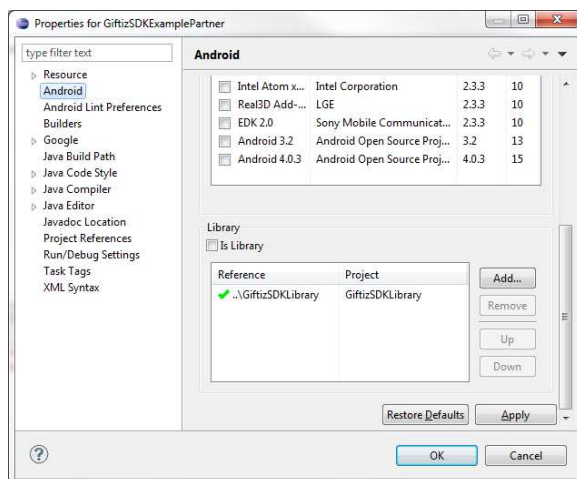
## 3. Step 1 / 4 – Link your project with Giftiz SDK Library



Giftiz SDK is an Android Library.

To link your project to it, just import GiftizSDKLibrary to your workspace.

And link your app to GiftizSDKLibrary:



**O  
P  
T  
I  
O  
N  
A  
L**

*We heartily recommend you use the Android Library as it helps you to keep your resources separated from the resources of Giftiz.*

*But if you don't want to use the Android Library, you can copy all the elements of **GiftizSDKLibrary/res/** folder to your App res/ folder and copy the jar in **GiftizSDKLibrary/libs/** to your App libs/ folder.*



## 4. Step 2 / 4 – Update the AndroidManifest.xml of your App

### a) Permissions

Add the following permissions if needed:

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

### b) Application – Partner Key and GiftizWebViewActivity

In the <application> part, add the two following entries:

```
<application ...>
  ...
  <meta-data android:name="GIFTIZ_PARTNER_KEY" android:value="YOUR_PARTNER_KEY"/>
  <activity android:name="com.purplebrain.giftiz.sdk.GiftizWebViewActivity" />
</application>
```

**YOUR\_PARTNER\_KEY** is a key specific to your app. When you're ready we'll send you one. It is a string like this one: 0ccd76b9-529c-4565-81ae-661721c3e0fa

The Giftiz SDK is only active if you also installed the Giftiz app on the device and created a Giftiz account.

For testing purposes, you can use the following key: **TEST\_PARTNER\_KEY**.

This test key activates the SDK (see Giftiz banners, mission complete and inApp purchases) even if you don't have a Giftiz account.

O  
P  
T  
I  
O  
N  
A  
L

### c) Application – Logs

To see Giftiz SDK Logs, you can add the following entry.

**Remove it before publishing to Google Play.**

```
<application ...>
  ...
  <meta-data android:name="SHOW_GIFTIZ_LOG" android:value="true" />
</application>
```

## 5. Step 3 / 4 – Call SDK Methods

### a) onResumeMainActivity + onPauseMainActivity

Add a call to the SDK in the onResume() method of your main screen activity. The parameter is the current activity, usually *this*.

```
@Override
protected void onResume() {
    GiftizSDK.onResumeMainActivity(this);
    ...
}
```



Do the same for `onPause()`:

```
@Override
protected void onPause() {
    GiftizSDK.onPauseMainActivity(this);
    ...
}
```

### b) missionComplete

Add a call to the SDK when the user completes the mission. The parameter is the current activity, usually `this`. Please send us a mail to tell us the name of mission you set.

```
GiftizSDK.missionComplete(this);
```

### c) InApp purchases

Add a call to the SDK when the user makes a purchase in your app.

```
float amountPaidByUser = 4.99f;
GiftizSDK.inAppPurchase(this, amountPaidByUser);
```

The first parameter is the current activity, usually `this`.

The second parameter is the amount in **Euros** spent by the user. Above is an example for an inApp of 4,99 €.

If you only work with other currencies, convert the amount to its value in euros and call the SDK with the euro amount.

Example 1: with £10 (£1 = 1,24 €), call SDK with **8.06f** ( $10/1.24 = 8.06$ ).

Example 2: with US\$10 (\$1 = 1.30€), call SDK with **7.69f** ( $10/1.3 = 7.69$ ).

Consider **the amount paid by the user** and not the amount you receive.

Please be careful with Google Play managed in app purchases and **don't call Giftiz SDK when you restore** previously bought in app purchases.

**PLEASE VERIFY THAT YOU ONLY CALL SDK  
WHEN THE USER REALLY MADE A PURCHASE**

### d) Paid Apps

The player will automatically earn Giftiz on the first launch of the app. You don't have anything to do.

## 6. Step 4 / 4 – Add the Giftiz Button on your main menu screen

Add the following View to the xml layout of your main screen activity:

```
<com.purplebrain.giftiz.sdk.GiftizButtonView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```




O  
P  
T  
I  
O  
N  
A  
L



O  
P  
T  
I  
O  
N  
A  
L

If you'd like, you can change the Giftiz button images by replacing the 3 following files in GiftizSDKLibrary:

- GiftizSDKLibrary/res/drawable-hdpi/giftiz\_logo.png
- GiftizSDKLibrary/res/drawable-hdpi/giftiz\_logo\_badge.png
- GiftizSDKLibrary/res/drawable-hdpi/giftiz\_logo\_warning.png

You'll find the badge , the warning  and the Giftiz Logo  to help you create the 3 images on the **Customize/** folder of the SDK. There are also 2 other kinds of already made buttons.

## 7. Congratulations!

If everything is ok so far for you, the Giftiz SDK is successfully integrated!

Please contact us ([etienne@giftiz.com](mailto:etienne@giftiz.com)) and send an apk of your app. Please tell us the name of the mission you set.



## 8. If you don't want to use the GiftizButtonView

Because of technology limitations, you may want to add the button in your game yourself and handle the interactions.

To do so, we provide you low-level functions with `GiftizSDK.Inner`. You can always refer to `GiftizSDKExamplePartner_SelfManaged` example project for details.

### a) Button Status

In order to know the status of the Giftiz button, call the following method:

```
GiftizSDK.Inner.getButtonStatus(this);
```

The parameter is the current activity, usually `this`. Call this method each time you load a screen with the Giftiz Button on it in order to know which button should be displayed.

Status can be:

- **ButtonInvisible** => Hide button
- **ButtonNaked** => `giftiz_logo.png`
- **ButtonBadge** => `giftiz_logo_badge.png`
- **ButtonWarning** => `giftiz_logo_warning.png`

### b) Button Clicked

To inform us that the user has clicked on the button, call the following method:

```
GiftizSDK.Inner.buttonClicked(this);
```

The parameter is the current activity, usually `this`.

### c) Button status listener

If the button needs to update its state while the screen is displayed, you need to be notified to change the image correctly.

To do so, register yourself as a delegate:

```
GiftizSDK.Inner.setButtonNeedsUpdateDelegate(delegate);
```

The delegate (=listener) parameter is your class. It should implement the following interface:

```
public interface ButtonNeedsUpdateDelegate {  
    public void buttonNeedsUpdate();  
}
```

After you set yourself as a delegate, whenever the button needs to be updated when already on screen, the SDK will call your `buttonNeedsUpdate()` method.

This **c)** part isn't mandatory as usually the button doesn't need to be updated live. But if you can handle it, you should do it. Moreover, when the user will click on the button, he'll see the badge (1) disappear right away.