Tabatoo»

A New Breed of Mobile App Monetization »

Tabatoo Monetization Sidebar » Get up to $10 CPM Extra! »

Tabatoo is a new breed of monetization network that works side by side with your current monetization tools. The Tabatoo Monetization Sidebar creates new assets on your App that are used for monetization, engagement and app distribution. It is non-intrusive and contributes to the experience and satisfaction from your app. It simply works!

Using Tabatoo can generate you up to $10 eCPM from your app (per 1000 app visits). Those revenues are added on top of your current revenues from banners, interstitials, in-app-purchase etc.

Engagement Widgets Are Included » Enhance Your App »

The Tabatoo Monetization Sidebar comes with out of the box engagement widgets that helps you make users stay for longer in your app and communicate better with their friends, with other app users and with you. All you have to do is go to the provided admin panel and pick the widgets that work best for you.

Monetization Sources » Search and Downloads »

- Your revenues are based on actual transactions:
  - A Search bar – enables your users to search the web and for you it generates mobile search based revenues.
  - Featured Apps – enable your users to enjoy the best of new apps that are targeted to them and for you to enjoy CPI based revenues
  - Courtesy Widgets like PayPal Donate are available for apps where appropriate

Adding Tabatoo to BuzzTouch on Android » Preparations »

- Adding Tabatoo to a BuzzTouch app can't be simpler and will take you only few minutes
- Before starting this short process, download the BuzzTouch app code to your development environment and make sure it builds and runs on the emulator. The instructions here are made for Eclipse IDE
- Sign up to Tabatoo here. Log-in to the admin panel and download the Android SDK from here. You can find your Tabatoo App ID here (search for "Tabatoo App ID" on page if you can't find it).
- You are now few tiny steps away from implementing Tabatoo »

1. Extract Tabatoo Android SDK to temporary folder
2. Copy the Tabatoo Jar file to the libs folder of the Android project. (If this folder does not exist, create one at the root of the Project)
   Use the Project properties to add the Jar to the build path (Eclipse Project/Properties/Java Build Path/ Libraries/ Add Jar File/ Tabatoo.jar)

3. A BuzzTouch Android app consists of few activities, in few files. You should add the Tabatoo code either BT_activity_base or BT_activity_root_tabs, depending on which layout you use in your app:
   3.1 Add the following code at the head of the file, importing the Tabatoo package and necessary packages as follows:

```
//Tabatoo
import android.view.View;
import android.view.ViewGroup;
import android.view.KeyEvent;
import android.widget.Button;
import com.tabatoo.slidingbar.Tabatoo;
import com.tabatoo.slidingbar.widget.MultiDirectionSlidingDrawer;
```

3.2 In each activity class, just before the BuzzTouch inserted text “//activity life-cycle events” , add the following variables:

```
//Tabatoo
MultiDirectionSlidingDrawer mDrawer;
Button button;
Tabatoo tabatoo;
//////////////////////////////////////////////////////////////////////
//activity life-cycle events.
```

3.3 At the end of the onCreate method (you can find it right after the after the “//activity life-cycle events” text) add the following code that creates and initializes the Tabatoo object as follows. Don’t forget to change “YOUR_APP_ID” to your actual App ID. You can find your Tabatoo App ID here (search for “Tabatoo App ID” on page if you can’t find it).

```
//Tabatoo
tabatoo = new Tabatoo(this,"YOUR_APP_ID"); //change this to your Tabatoo App ID
View tabatoo_view =
((ViewGroup)getWindow().getDecorView()).findViewById(android.R.id.content);
mDrawer = tabatoo.initDrawer(tabatoo_view);
```

3.4 An example to how the code will look like is as follows (Tabatoo code in Purple):

```
package com.example1.example2;
import static com.schoolshortcut.BT_gcmConfig.EXTRA_MESSAGE;
//More code…
import com.google.android.gcm.GCMRegistrar;

//Tabatoo
import android.view.View;
import android.view.ViewGroup;
import android.view.KeyEvent;
import android.widget.Button;
import com.tabatoo.slidingbar.Tabatoo;
import com.tabatoo.slidingbar.widget.MultiDirectionSlidingDrawer;

public class BT_activity_base extends Activity implements LocationListener{
	//More code…
	//visible / showing
	public static final int INVISIBLE = 4;
	public static final int VISIBLE = 0;

	//Tabatoo
	MultiDirectionSlidingDrawer mDrawer;
	Button button;
	Tabatoo tabatoo;
	////////////////////////////////////////////////////////////////////
	//activity life-cycle events.
	//onCreate
	@Override
	public void onCreate(Bundle savedInstanceState){

	//onCreate Code…

	//Tabatoo
	tabatoo = new Tabatoo(this,“YOUR_APP_ID”);
	View  tabatoo_view = ((ViewGroup) getWindow().getDecorView()).findViewById(android.R.id.content);
	mDrawer = tabatoo.initDrawer(tabatoo_view);

}
//onStart
@Override
protected void onStart() {
```

4. For each activity class, add the following code, implementing the onKeyDown method and revise the onBackPressed() methods to include a call to Tabatoo when the back key is pressed. As follows:

```
//Tabatoo – add this function to monitor the back key
 @Override
 public boolean onKeyDown(int keyCode, KeyEvent event)  {
      if (android.os.Build.VERSION.SDK_INT < android.os.Build.VERSION_CODES.ECLAIR
            && keyCode == KeyEvent.KEYCODE_BACK
            && event.getRepeatCount() == 0) {

          // Take care of calling this method on earlier versions of
          // the platform where it doesn't exist.
 onBackPressed();
          }
          return super.onKeyDown(keyCode, event);
   }

//Tabatoo -  revise the onBackPressed code to call Tabatoo when pressing the back key:
@Override
   public void onBackPressed() {
          BT_debugger.showIt(activityName + ":onBackPressed");
          BT_debugger.showIt(activityName + ":REVERSING TRANSITIONS ARE DISABLED");

          mDrawer.onBackPressed(this);
 }
```

5. In the AndroidManifest.xml:

   5.1 add the following activity reference:

```
<!-- remaining activities are for individual plugin types -->
<activity android:name="com.tabatoo.slidingbar.TabatooBrowserActivity"
android:icon="@drawable/icon" android:label="@string/app_name"/>
```

   5.2 add if do not exist already, the following network permissions:

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

That’s it. You are done!

FAQ:

1. Q: How do I remove the handle and leave only the button as a mean to open/close the bar (or in other words, how do I toggle the handle to be shown/hidden)?
   A: Use the following command:

```
mDrawer.showHadler(); //show the handle
mDrawer.hideHandler(); //hide the handle
```

2. Q: How do I open/close the bar programmatically?
   A: Use the following commands:

```
mDrawer.open();  //expose the bar
mDrawer.close();  //hide the bar
```

3. Q: How do I change the position of the handle to be on the upper or lower part of the screen?
   A: The default location of the handle is in the center of the screen. Use the following commands to change it on the fly:

```
mDrawer.setAlignHandler(Tabatoo.TOP);  //align the handle at the top of the screen
mDrawer.setAlignHandler(Tabatoo.CENTER);  //align the handle at the center of the screen
mDrawer.setAlignHandler(Tabatoo.BOTTOM);  //align the handle at the bottom of the screen
```

4. Q: How do I change the shape of the handle?
   A: In order to change the default shapes of the handle use the following commands. Note that there are two assets you should add to the project. The first for the handle when the bar is closed. The second for when the bar is opened. Obviously, those two can be the same. The size of the provided assets should fit medium density devices. The SDK will scale it in/out as appropriate.

```
mDrawer.setHandlerDrawableClose(getResources().getDrawable(R.drawable.asset_name_close
d));  // set a custom handle image for a hidden/closed bar
mDrawer.setHandlerDrawableOpen(getResources().getDrawable(R.drawable.asset_name_open
ed)); // set a custom handle image for an exposed/opened bar
```

5. Q: What if I want the bar to open from right to left and not from left to right
   A: No prob. Tabatoo supports bar exposure from left and from right. While left to right is the default, you can change it to appear on the right and open to the left. Please note that in this case, Tabatoo will flip the handle as well. Meaning the default handle, or your custom handle will be mirrored to reflect a right to left transition.

   All you have to do it initialize the Tabatoo drawer with the following parameter:

```
mDrawer = tabatoo.initDrawer(tabatoo_view, Tabatoo.RIGHT);
```