Getting Started for Android

Welcome to Android game development with the Google Play game services!

The Google Play game services SDK provides cross-platform game services that lets you easily integrate popular gaming features such as achievements, leaderboards, Cloud Save, and real-time multiplayer (on Android) in your tablet and mobile games.

This training will guide you to install a sample game application for Android and quickly get started to create your own Android game. The Type-a-Number Challenge sample app demonstrates how you can integrate achievements and leaderboards into your game.

Before you begin

- You should have your Android development environment set up. If you are new to developing Android applications, see Building Your First App.
- You should have a physical device running Android 2.2 (Froyo) or higher for testing.

Step 1: Download the sample app

For this training, you will need to download the Type-a-Number Challenge sample Android application.

To download the sample application:

1. Download the Android samples by following the link in the Downloads page, then extract the sample files to your development machine.
2. Install the Google Play services SDK.
3. Import the `TypeANumber` and `BaseGameUtils` code from the Android samples into your workspace. To do this in Eclipse:
   a. Click **File** > **Import** > **Android** > **Existing Android Code into Workspace**.
   b. Browse to the directory where you downloaded `TypeANumber` on your development machine and click **OK** to import the files.
   c. Perform the same steps for `BaseGameUtils`.
4. Make sure that the Google Play services and `BaseGameUtils` library projects are referenced in your Android project. To do this in Eclipse:
   a. Right-click your Android project and select **Properties** > **Android**.
   b. In the **Library** section, add the `google-play-services-lib` project and the `BaseGameUtils` project, then click **OK** to save.
5. In the Type-a-Number Challenge project (not `BaseGameUtils`), open `AndroidManifest.xml` and change the package name from `com.google.example.games.tanc` to a different package name of your own. The new package name must not start with `com.google`, `com.example`, or `com.android`.
6. Correct the references to the `R` resource object in the source files. To do this, add `import your.package.name.R;` to each source file that references the `R` object (replace `your.package.name` with the actual package name).

Step 2: Set up the game in the Developer Console

The Google Play Developer Console is where you manage game services for your game, and configure metadata for authorizing and authenticating your game.

To set up the sample game in the Developer Console:

1. Point your web browser to the Developer Console, and sign in. If you haven't registered for the Developer Console before, you will be prompted to do so.
2. Follow these instruction to add your game to the Developer Console.
   a. When asked if you use Google APIs in your app, select **I don't use any Google APIs in my game yet**.
   b. For the purpose of this training, you can fill up the form with your own game details. For convenience, you can use the placeholder icons and screenshots provided in the Downloads page.
3. Follow these instructions to generate an OAuth 2.0 client ID for your Android app.
   a. When linking your Android app, make sure to specify the exact package name you used previously when renaming sample package.
   b. You can use the Eclipse Export Wizard to generate a new keystore and signed certificate if you don't have one already. To learn how to run the Export Wizard, see Compile and sign with Eclipse ADT.
4. Make sure to record the following information for later:
   a. Your application ID: This is a string consisting only of digits (typically 12 or more), at the beginning of your client ID.
   b. Your signing certificate: Note which certificate you used when setting up your API access (the certificate whose SHA1 fingerprint you provided). You should use the same certificate to sign your app when testing or releasing your app.
5. Configure achievements for Type-a-Number Challenge:
   a. Select the **Achievements** tab in the Developer Console.
   b. Add the following sample achievements:

Name	Description	Special Instructions
Prime	Get a score that's a prime number.	*None*
Humble	Request a score of 0.	*None*
Don't get cocky, kid	Request a score of 9999 in either mode.	*None*
OMG U R TEH UBER LEET!	Receive a score of 1337.	Make this a hidden achievement.
Bored	Play the game 10 times.	Make this an an incremental achievement with 10 steps to unlock.
Really Really Bored	Play the game 100 times.	Make this an an incremental achievement with 100 steps to unlock.

   c. Record the IDs (long alphanumeric strings) for each achievement that you created.
   d. Configure achievements that are appropriate for your game. To learn more, see the concepts behind achievements and how to implement achievements in Android.
6. Configure the leaderboards for Type-a-Number Challenge:
   a. Select the the **Leaderboards** tab in the Developer Console.
   b. Add two sample leaderboards: one named "Easy High Scores" and another named "Hard High Scores". Both leaderboards should use Integer score formatting with 0 decimal places, and an ordering type of **Larger is better**.
   c. Record the IDs (long alphanumeric strings) for each leaderboard you created.
   d. Configure leaderboards that are appropriate for your game. To learn more, see the concepts behind leaderboards and how to implement leaderboards in Android.
7. Add test accounts for your game. This step is needed only for apps that have not yet been published in the Developer Console. Before the app is published, only the test accounts listed in the Developer Console can log in. However, once an application is published, everyone is allowed to log in.

> **Warning:** If you try to make Google Play game services SDK calls for an unpublished game by using an account that's not listed as a test account, the game services will behave as if the game did not exist and you'll get back the `ConnectionResult.SIGN_IN_REQUIRED` return code. If you attempt to launch `ConnectionResult.startResolutionForResult()`, you'll get back `GamesActivityResultCodes.RESULT_SIGN_IN_FAILED`.

Step 3: Modify your code

To run the game, you need to configure the application ID as a resource in your Android project. You will also need to add games metadata in the `AndroidManifest.xml`.

1. Open `res/values/ids.xml` and replace the placeholder IDs. If you are creating an Android game

from scratch, you will need to create this file first.

   a. Specify your application ID in the `app_id` resource.
   b. Specify each achievement ID that you created earlier in the corresponding `achievement_*` resource.
   c. Specify each leaderboard ID that you created earlier in the corresponding `leaderboard_*` resource.
2. Open `AndroidManifest.xml` and enter your package name in the `package` attribute of the <`manifest`> element. If you are creating an Android game from scratch, make sure that you also add the following code inside the <`application`> element:

```
<meta-data android:name="com.google.android.gms.games.APP_ID"
     android:value="@string/app_id" />
```

Step 4: Test your game

To ensure that game services are functioning correctly in your game, test the application before you publish it on Google Play.

> **Note:** It's recommended that you test on a physical Android device. However, if you do not have a physical device, you can test against the Android Emulator. To do so, download the emulator system image that includes the Google Play Services, under **Android 4.2.2**, from the SDK Manager.

To run your game on your physical test device:

1. Verify that you have set up the test account that you are using to log in to the app (as described in Step 2).
2. Export an APK and sign it with the same certificate that you used to set up the project in Developer Console. To export a signed APK in Eclipse, click **File** > **Export** > **Android** > **Export Android Application**.
3. Install the signed APK on your physical test device by using the `adb` tool. To learn how to install an application, see Running on a Device.

> **Warning:** When you run the application directly from Eclipse, Eclipse will sign the application with your debug certificate by default. If you did not use this debug certificate when setting up the application in Developer Console, this will cause errors. Make sure to run an APK that you exported and signed with a certificate that matches one of the certificates you used during the application setup in Developer Console.

Next steps

Learn more about using game services with Android:

- Initialize your `GamesClient`
- Add achievements to your game
- Build your own leaderboards
- Save games in the cloud
- Understand how anti-piracy works
- Develop real-time multiplayer games
- Enable verbose logging

Last updated May 15, 2013.