Google+ Platform
Quickstart
PhotoHunt Sample
API Reference
Features
Android
Getting started
Sign in
Share
+1 button
Over-the-air installs
App activities
People and profiles
Google Play services
iOS
Websites
Hangout Apps
Downloads
Best Practices
Branding Guidelines
Community & Support
Release Notes
Google APIs Console
Developer Policies
Terms of Service

Getting Started with the Google+ Platform for Android

Before you can start integrating Google+ features in your own app, you must create a Google APIs Console project and initialize the `PlusClient` within your app. You can try out the quick-start sample application if you would like to quickly experiment with the platform.

Prerequisites
Step 1: Enable the Google+ API
Step 2: Configure your Eclipse project
Step 3: Initialize the PlusClient
Next steps
Frequently asked questions

Prerequisites

The Google+ platform for Android has the following requirements:

- A physical device to use for developing and testing because Google Play services can only be installed on an emulator with an AVD that runs Google APIs platform based on Android 4.2.2 or higher.
- The latest version of the Android SDK, including the SDK Tools component. The SDK is available from the **Android SDK Manager**.
- Your project to compile against Android 2.2 (Froyo) or higher.
- Eclipse configured to use Java 1.6
- The Google Play Services SDK:
  1. Launch Eclipse and select **Window** > **Android SDK Manager** or run `android` from the command line.
  2. Scroll to the bottom of the package list and select **Extras** > **Google Play services**. The package is downloaded to your computer and installed in your SDK environment at `<android-sdk-folder>/extras/google/google_play_services`.

Step 1: Enable the Google+ API

If you already ran the quick start, the following steps are similar; however, this time you will create a OAuth 2.0 client for your package rather than for the quick-start package.

To authenticate and communicate with the Google+ APIs, you must first register your digitally signed `.apk` file's public certificate in the Google APIs Console:

1. In the Google APIs Console, create an API project for your application.
2. In the Services pane, enable the **Google+ API** and any other APIs that your app requires.
3. In the API Access pane, create an OAuth 2.0 client ID by clicking **Create an OAuth 2.0 Client ID** :
   a. Type a product name in the dialog box that displays, and click **Next**. Providing a product logo and home page URL are optional.
   b. Choose **Installed application** as your **Application type** and select **Android** as type.
   c. In the **Package name** field, enter your Android's app's package name.
   d. In a terminal, run the the Keytool utility to get the SHA-1 fingerprint of the certificate. For the `debug.keystore`, the password is **`android`**.

```
keytool -exportcert -alias androiddebugkey -keystore <path-to-debug-or-production-
```

> **Note:** For Eclipse, the debug keystore is typically located at *~/.android/debug.keystore*.

The Keytool prints the fingerprint to the shell. For example:

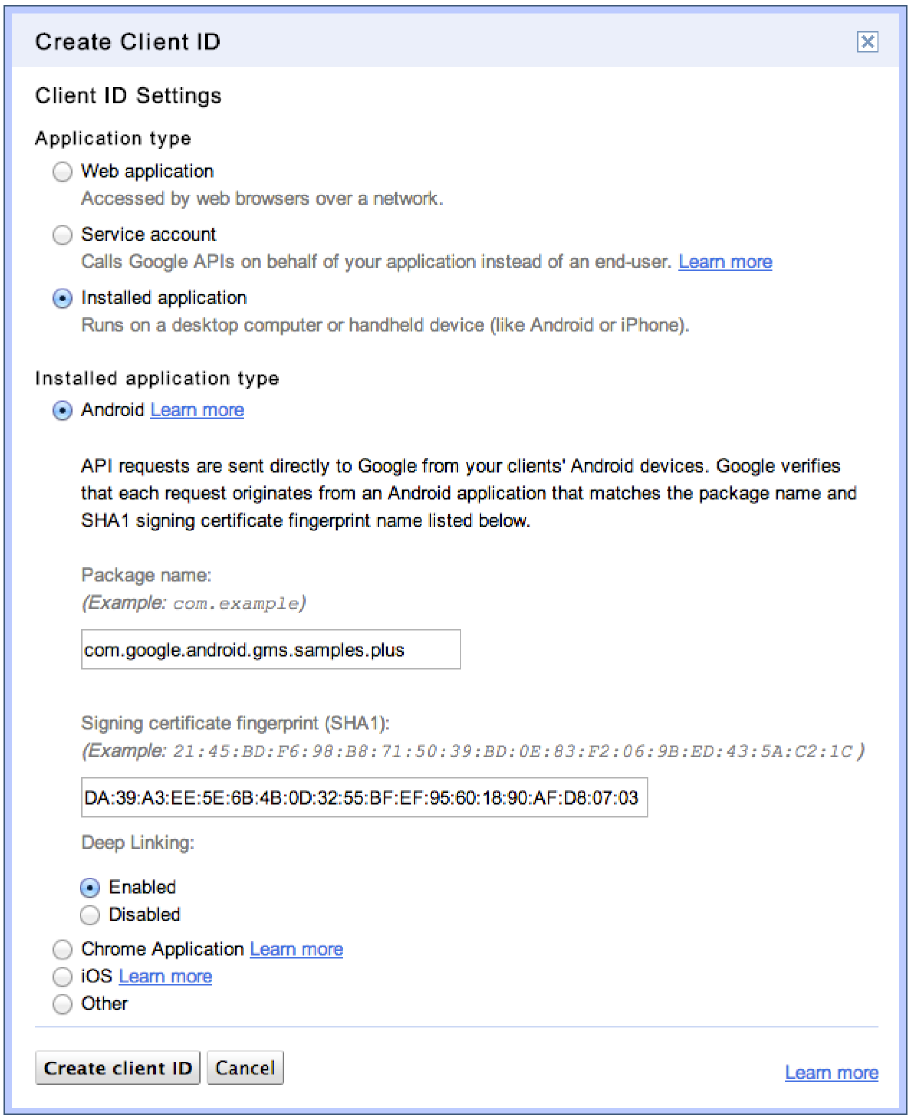
```
$ keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore -
Enter keystore password: Type "android" if using debug.keystore
Alias name: androiddebugkey
Creation date: Aug 27, 2012
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 503bd581
Valid from: Mon Aug 27 13:16:01 PDT 2012 until: Wed Aug 20 13:16:01 PDT 2042
```

```
Certificate fingerprints:
     MD5:  1B:2B:2D:37:E1:CE:06:8B:A0:F0:73:05:3C:A3:63:DD
     SHA1: D8:AA:43:97:59:EE:C5:95:26:6A:07:EE:1C:37:8E:F4:F0:C8:05:C8
     SHA256: F3:6F:98:51:9A:DF:C3:15:4E:48:4B:0F:91:E3:3C:6A:A0:97:DC:0A:3F:B2:D2:E1
     Signature algorithm name: SHA1withRSA
     Version: 3
```

Copy the SHA1 hash, which is highlighted in the example above.

> **Important:** When you prepare to release your app to your users, you will follow these steps again and create a new OAuth 2.0 client ID for your production app. For production apps, you will use your own private key to sign the production app's `.apk` file. See Signing your applications for more information.

e. Paste the **SHA-1** fingerprint into the **Signing certificate fingerprint** field.
f. To activate interactive posts, enable the **Deep Linking** option.


Create Client ID

Client ID Settings

Application type

Web application [✓ Installed application] Service account

Accessed by web browsers over a network.

Calls Google APIs on behalf of your application instead of an end-user. Learn more

Runs on a desktop computer or handheld device (like Android or iPhone).

Installed application type

[✓ Android] Learn more

API requests are sent directly to Google from your clients' Android devices. Google verifies that each request originates from an Android application that matches the package name and SHA1 signing certificate fingerprint name listed below.

Package name:
(Example: com.example)

com.google.android.gms.samples.plus

Signing certificate fingerprint (SHA1):
(Example: 21:45:BD:F6:98:B8:71:50:39:BD:0E:83:F2:06:9B:ED:43:5A:C2:1C)

DA:39:A3:EE:5E:6B:4B:0D:32:55:BF:EF:95:60:18:90:AF:D8:07:03

Deep Linking:

[✓ Enabled] Disabled

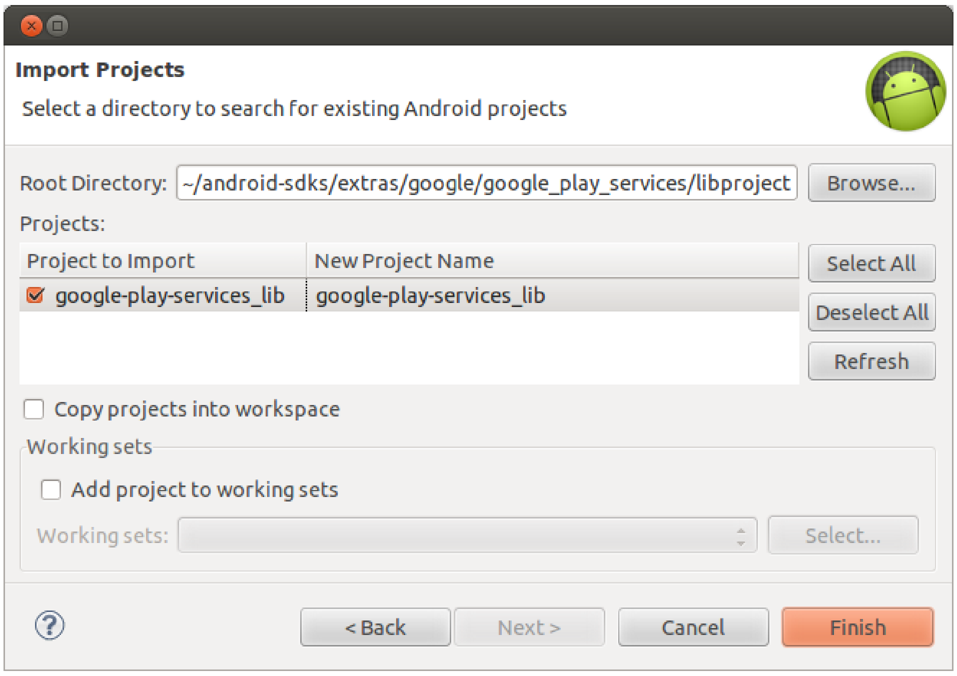
Chrome Application Learn more
iOS Learn more
Other

g. Click the **Create client ID** button.

Step 2: Configure your Eclipse project

Integrating Google+ features in your Android app requires configuring your Android project in Eclipse. You will import and reference the Google Play services library project and set up the Java build path and libraries.

1. Launch Eclipse.
2. Select **File** > **Import** > **Android** > **Existing Android Code Into Workspace** and click **Next**.
3. Select **Browse...**. Enter `<android-sdk-folder>/extras/google/google_play_services/libproject`.


Import Projects

Select a directory to search for existing Android projects

Root Directory: ~/android-sdks/extras/google/google_play_services/libproject

Projects:

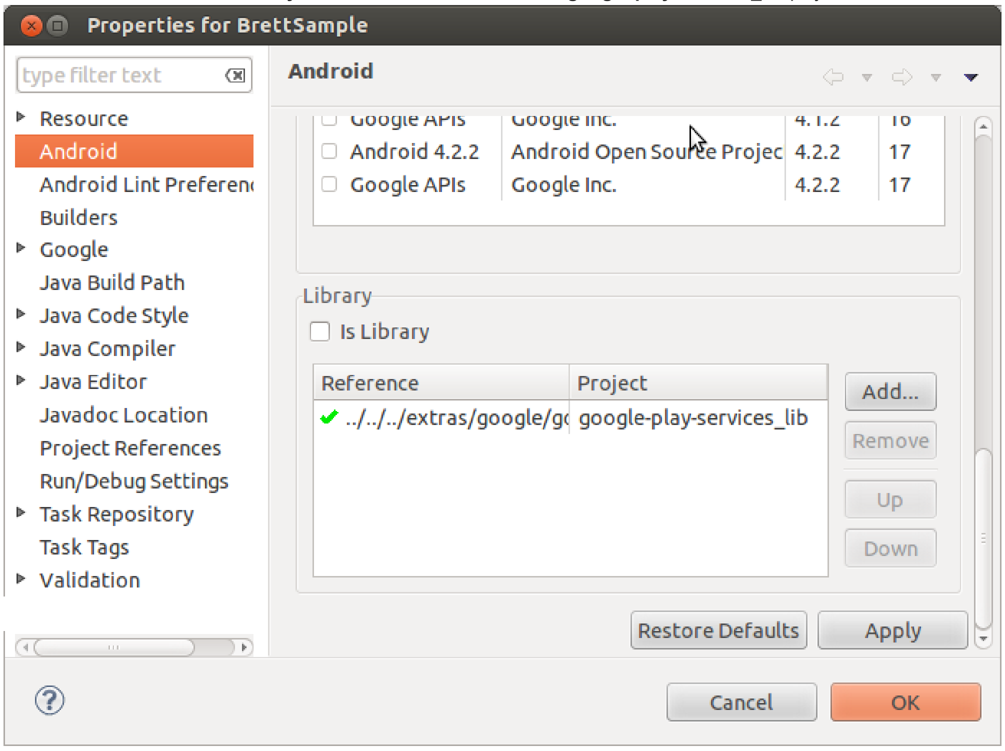
Project to Import	New Project Name
✓ google-play-services_lib	google-play-services_lib

Copy projects into workspace

Working sets

Add project to working sets

4. Update the your project's properties:
   a. Click **Project** > **Properties**. The project properties dialog displays.
   b. Select **Android**, and in the **Library** section, click **Add**. Choose the **google-play-services_lib** project.


Properties for BrettSample

Android

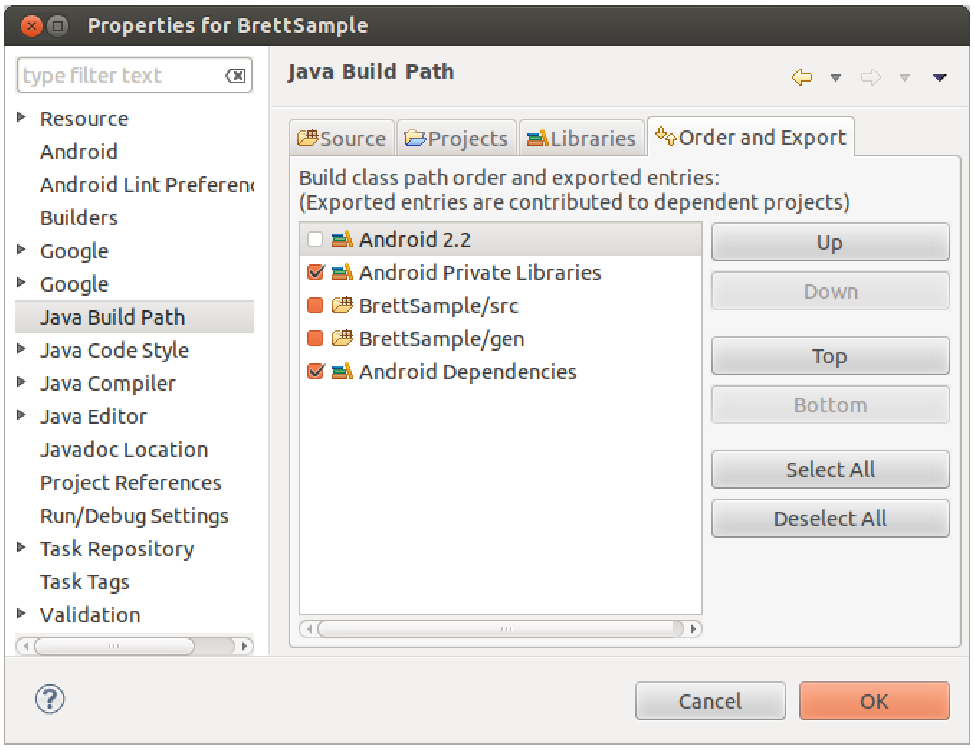
Google APIs	Google Inc.	4.1.2	16
Android 4.2.2	Android Open Source Projec	4.2.2	17
Google APIs	Google Inc.	4.2.2	17

Library

Is Library

Reference	Project
✓ ../../../extras/google/gc	google-play-services_lib

   c. Select **Java Build Path**, and click the **Order and Export** tab, select `Android Private Libraries`.


Properties for BrettSample

Java Build Path

Build class path order and exported entries:
(Exported entries are contributed to dependent projects)

- [ ] Android 2.2
- [x] Android Private Libraries
- [x] BrettSample/src
- [x] BrettSample/gen
- [x] Android Dependencies

d. Click **OK**.

5. Optional: Click **Project** > **Clean** to ensure your project picks up the settings.

Important: You will require a physical device when you run your project. The Google Play services library requires a physical device for testing projects on Android 4.2.2 or earlier.

Step 3: Initialize the PlusClient

The `PlusClient` object is used to communicate with the Google+ service and becomes functional after an asynchronous connection has been established with the service. Because the client makes a connection to a service, you want to make sure the `PlusClient.disconnect` method is called when appropriate to ensure robustness.

Typically, you want to manage the `PlusClient` in your activity's lifecycle.

- Initialize the `PlusClient` object in your Activity.onCreate handler.
- Invoke `PlusClient.connect` during Activity.onStart.
- Invoke `PlusClient.disconnect` during Activity.onStop.

Your activity will listen for when the connection has established or failed by implementing the `ConnectionCallbacks` and `OnConnectionFailedListener` interfaces.

```
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GooglePlayServicesClient.ConnectionCallbacks;
import com.google.android.gms.common.GooglePlayServicesClient.OnConnectionFailedListener;

public class ExampleActivity extends Activity implements
        ConnectionCallbacks, OnConnectionFailedListener {
    private static final int REQUEST_CODE_RESOLVE_ERR = 9000;

    private ProgressDialog mConnectionProgressDialog;
    private PlusClient mPlusClient;
    private ConnectionResult mConnectionResult;
```

When the `PlusClient` object is unable to establish a connection, your implementation has an opportunity to recover inside your implementation of `onConnectionFailed`, where you are passed a connection status that can be used to resolve any connection failures. You should save this connection status in a member variable and invoke it by calling `ConnectionResult.startResolutionForResult` when the user presses the sign-in button or +1 button.

```
@Override
public void onConnectionFailed(ConnectionResult result) {
       if (mConnectionProgressDialog.isShowing()) {
               // The user clicked the sign-in button already. Start to resolve
               // connection errors. Wait until onConnected() to dismiss the
               // connection dialog.
```

```
                if (result.hasResolution()) {
                        try {
                                result.startResolutionForResult(this, REQUEST_CODE_RESOLVE_E
                        } catch (SendIntentException e) {
                                mPlusClient.connect();
                        }
                }
        }

        // Save the intent so that we can start an activity when the user clicks
        // the sign-in button.
        mConnectionResult = result;
}

@Override
public void onConnected(Bundle connectionHint) {
 // We've resolved any connection errors.
  mConnectionProgressDialog.dismiss();
}
```

Because the resolution for the connection failure was started with `startActivityForResult` and the code `REQUEST_CODE_RESOLVE_ERR`, we can capture the result inside Activity.onActivityResult.

```
@Override
protected void onActivityResult(int requestCode, int responseCode, Intent intent) {
    if (requestCode == REQUEST_CODE_RESOLVE_ERR && responseCode == RESULT_OK) {
        mConnectionResult = null;
        mPlusClient.connect();
    }
}
```

After adding the `PlusClient` to the lifecycle of your Activity, your class should resemble the following structure:

```
import com.google.android.gms.common.*;
import com.google.android.gms.common.GooglePlayServicesClient.*;
import com.google.android.gms.plus.PlusClient;

public class ExampleActivity extends Activity implements View.OnClickListener,
        ConnectionCallbacks, OnConnectionFailedListener {
    private static final String TAG = "ExampleActivity";
    private static final int REQUEST_CODE_RESOLVE_ERR = 9000;

    private ProgressDialog mConnectionProgressDialog;
    private PlusClient mPlusClient;
    private ConnectionResult mConnectionResult;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mPlusClient = new PlusClient.Builder(this, this, this)
                .setVisibleActivities("http://schemas.google.com/AddActivity", "http://sche
                .build();
        // Progress bar to be displayed if the connection failure is not resolved.
        mConnectionProgressDialog = new ProgressDialog(this);
        mConnectionProgressDialog.setMessage("Signing in...");
    }

    @Override
    protected void onStart() {
        super.onStart();
        mPlusClient.connect();
    }

    @Override
    protected void onStop() {
        super.onStop();
        mPlusClient.disconnect();
    }

    @Override
    public void onConnectionFailed(ConnectionResult result) {
        if (result.hasResolution()) {
            try {
                result.startResolutionForResult(this, REQUEST_CODE_RESOLVE_ERR);
            } catch (SendIntentException e) {
                mPlusClient.connect();
            }
        }
        // Save the result and resolve the connection failure upon a user click.
        mConnectionResult = result;
    }

    @Override
    protected void onActivityResult(int requestCode, int responseCode, Intent intent) {
        if (requestCode == REQUEST_CODE_RESOLVE_ERR && responseCode == RESULT_OK) {
            mConnectionResult = null;
            mPlusClient.connect();
        }
    }

    @Override
    public void onConnected(Bundle connectionHint) {
        String accountName = mPlusClient.getAccountName();
        Toast.makeText(this, accountName + " is connected.", Toast.LENGTH_LONG).show();
    }

    @Override
    public void onDisconnected() {
        Log.d(TAG, "disconnected");
    }
}
```

Next steps

Now that you have added `PlusClient` to your activity, you can integrate your choice of Google+ features to your app:

- Sign in with Google
- Create interactive posts
- Manage app activities
- Fetch people information

Frequently asked questions

How do I continue to support older platforms, such as Eclair (v2.0)?

Check system version at runtime, and only call `PlusClient` if the version is Froyo (v2.2) or higher.

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.FROYO) {
    // Make the call to PlusClient
}
```

How do I check to see if the Google+ app is installed on the device?

The `GooglePlusUtil#checkGooglePlusApp` utility method returns `GooglePlusUtil.SUCCESS` if the Google+ app is installed. If the app is not installed, you can prompt the user to install the app with the `GooglePlusUtil#getErrorDialog` utility method.

```
int errorCode = GooglePlusUtil.checkGooglePlusApp(this);
if (errorCode != GooglePlusUtil.SUCCESS) {
  GooglePlusUtil.getErrorDialog(errorCode, this, 0).show();
}
```

How do I debug my Google+ integration?

By enabling logging, you can diagnose network issues when working with the Google APIs.

To enable logging, run the following command:

```
adb shell setprop log.tag.GooglePlusPlatform VERBOSE
```

To disable logging, run the following command:

```
adb shell setprop log.tag.GooglePlusPlatform ""
```