Retrieving the Current Location

Location Services automatically maintains the user's current location, so all your app has to do is retrieve it as needed. The location's accuracy is based on the location permissions you've requested and location sensors that are currently active for the device.

Location Services sends the current location to your app through a location client, which is an instance of the Location Services class LocationClient (/reference/com/google/android/gms/location/LocationClient.html). All requests for location information go through this client.

THIS LESSON TEACHES YOU TO

YOU SHOULD ALSO READ

- Setup Google Play Services SDK

TRY IT OUT

Download the sample

LocationUpdates.zip

> **Note:** Before you start the lesson, be sure that your development environment and test device are set up correctly. To learn more about this, read the Setup (/google/play-services/setup.html) section in the Google Play services guide.

Specify App Permissions

Apps that use Location Services must request location permissions. Android has two location permissions: ACCESS_COARSE_LOCATION (/reference/android/Manifest.permission.html#ACCESS_COARSE_LOCATION) and ACCESS_FINE_LOCATION (/reference/android/Manifest.permission.html#ACCESS_FINE_LOCATION). The permission you choose controls the accuracy of the current location. If you request only coarse location permission, Location Services obfuscates the returned location to an accuracy that's roughly equivalent to a city block.

Requesting ACCESS_FINE_LOCATION (/reference/android/Manifest.permission.html#ACCESS_FINE_LOCATION) implies a request for ACCESS_COARSE_LOCATION (/reference/android/Manifest.permission.html#ACCESS_COARSE_LOCATION).

For example, to add ACCESS_COARSE_LOCATION (/reference/android/Manifest.permission.html#ACCESS_COARSE_LOCATION), insert the following as a child element of the <manifest> (/guide/topics/manifest/manifest-element.html) element:

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

Check for Google Play Services

Location Services is part of the Google Play services APK. Since it's hard to anticipate the state of the user's device, you should always check that the APK is installed before you attempt to connect to Location Services. To check that the APK is installed, call GooglePlayServicesUtil.isGooglePlayServicesAvailable() (/reference/com/google/android/gms/common/GooglePlayServicesUtil.html#isGooglePlayServicesAvailable(

android.content.Context)), which returns one of the integer result codes listed in the reference documentation for ConnectionResult (/reference/com/google/android/gms/common/ConnectionResult.html). If you encounter an error, call GooglePlayServicesUtil.getErrorDialog() (/reference/com/google/android/gms/common/GooglePlayServicesUtil.html#getErrorDialog(int, android.app.Activity, int)) to retrieve localized dialog that prompts users to take the correct action, then display the dialog in a DialogFragment (/reference/android/support/v4/app/DialogFragment.html). The dialog may allow the user to correct the problem, in which case Google Play services may send a result back to your activity. To handle this result, override the method onActivityResult() (/reference/android/support/v4/app/FragmentActivity.html#onActivityResult(int, int, android.content.Intent)).

Since you usually need to check for Google Play services in more than one place in your code, define a method that encapsulates the check, then call the method before each connection attempt. The following snippet contains all of the code required to check for Google Play services:

```
public class MainActivity extends FragmentActivity {
    ...
    // Global constants
    /*
     * Define a request code to send to Google Play services
     * This code is returned in Activity.onActivityResult
     */
    private final static int
            CONNECTION_FAILURE_RESOLUTION_REQUEST = 9000;
    ...
    // Define a DialogFragment that displays the error dialog
    public static class ErrorDialogFragment extends DialogFragment {
        // Global field to contain the error dialog
        private Dialog mDialog;
        // Default constructor. Sets the dialog field to null
        public ErrorDialogFragment() {
            super();
            mDialog = null;
        }
        // Set the dialog to display
        public void setDialog(Dialog dialog) {
            mDialog = dialog;
        }
        // Return a Dialog to the DialogFragment.
        @Override
        public Dialog onCreateDialog(Bundle savedInstanceState) {
            return mDialog;
        }
    }
    ...
    /*
     * Handle results returned to the FragmentActivity
     * by Google Play services
     */
    @Override
    protected void onActivityResult(
            int requestCode, int resultCode, Intent data) {
        // Decide what to do based on the original request code
        switch (requestCode) {
            ...
            case CONNECTION_FAILURE_RESOLUTION_REQUEST :
            /*
             * If the result code is Activity.RESULT_OK, try
             * to connect again
             */
```

```
                switch (resultCode) {
                    case Activity.RESULT_OK :
                    /*
                     * Try the request again
                     */
                    ...
                    break;
                }
            ...
        }
     }
    ...
    private boolean servicesConnected() {
        // Check that Google Play services is available
        int resultCode =
                GooglePlayServicesUtil.
                        isGooglePlayServicesAvailable(this);
        // If Google Play services is available
        if (ConnectionResult.SUCCESS == resultCode) {
            // In debug mode, log the status
            Log.d("Location Updates",
                    "Google Play services is available.");
            // Continue
            return true;
        // Google Play services was not available for some reason
        } else {
            // Get the error code
            int errorCode = connectionResult.getErrorCode();
            // Get the error dialog from Google Play services
            Dialog errorDialog = GooglePlayServicesUtil.getErrorDialog(
                    errorCode,
                    this,
                    CONNECTION_FAILURE_RESOLUTION_REQUEST);

            // If Google Play services can provide an error dialog
            if (errorDialog != null) {
                // Create a new DialogFragment for the error dialog
                ErrorDialogFragment errorFragment =
                        new ErrorDialogFragment();
                // Set the dialog in the DialogFragment
                errorFragment.setDialog(errorDialog);
                // Show the error dialog in the DialogFragment
                errorFragment.show(getSupportFragmentManager(),
                        "Location Updates");
            }
        }
    }
    ...
}
```

Snippets in the following sections call this method to verify that Google Play services is available.

Define Location Services Callbacks

To get the current location, create a location client, connect it to Location Services, and then call its `getLastLocation()` `(/reference/com/google/android/gms/location/LocationClient.html#getLastLocation())` method. The return value is the best, most recent location, based on the permissions your app requested and the currently-enabled location sensors.

Before you create the location client, implement the interfaces that Location Services uses to communicate with your app:

`ConnectionCallbacks`
: Specifies methods that Location Services calls when a location client is connected or disconnected.

`OnConnectionFailedListener`
: Specifies a method that Location Services calls if an error occurs while attempting to connect the location client. This method uses the previously-defined `showErrorDialog` method to display an error dialog that attempts to fix the problem using Google Play services.

The following snippet shows how to specify the interfaces and define the methods:

```
public class MainActivity extends FragmentActivity implements
        GooglePlayServicesClient.ConnectionCallbacks,
        GooglePlayServicesClient.OnConnectionFailedListener {
    ...
    /*
     * Called by Location Services when the request to connect the
     * client finishes successfully. At this point, you can
     * request the current location or start periodic updates
     */
    @Override
    public void onConnected(Bundle dataBundle) {
        // Display the connection status
        Toast.makeText(this, "Connected", Toast.LENGTH_SHORT).show();

    }
    ...
    /*
     * Called by Location Services if the connection to the
     * location client drops because of an error.
     */
    @Override
    public void onDisconnected() {
        // Display the connection status
        Toast.makeText(this, "Disconnected. Please re-connect.",
                Toast.LENGTH_SHORT).show();
    }
    ...
    /*
     * Called by Location Services if the attempt to
     * Location Services fails.
     */
    @Override
    public void onConnectionFailed(ConnectionResult connectionResult) {
        /*
         * Google Play services can resolve some errors it detects.
         * If the error has a resolution, try sending an Intent to
         * start a Google Play services activity that can resolve
         * error.
         */
        if (connectionResult.hasResolution()) {
            try {
                // Start an Activity that tries to resolve the error
                connectionResult.startResolutionForResult(
                        this,
                        CONNECTION_FAILURE_RESOLUTION_REQUEST);
                /*
                 * Thrown if Google Play services canceled the original
                 * PendingIntent
                 */
            } catch (IntentSender.SendIntentException e) {
                // Log the error
                e.printStackTrace();
            }
        } else {
            /*
             * If no resolution is available, display a dialog to the
             * user with the error.
             */
            showErrorDialog(connectionResult.getErrorCode());
        }
    }
    ...
}
```

Connect the Location Client

Now that the callback methods are in place, create the location client and connect it to Location Services.

You should create the location client in onCreate() (/reference/android/support/v4/app/FragmentActivity.html#onCreate(android.os.Bundle)), then connect it in onStart() (/reference/android/support/v4/app/FragmentActivity.html#onStart()), so that Location Services maintains the current location while your activity is fully visible. Disconnect the client in onStop() (/reference/android/support/v4/app/FragmentActivity.html#onStop()), so that when your app is not visible, Location Services is not maintaining the current location. Following this pattern of connection and disconnection helps save battery power. For example:

> **Note:** The current location is only maintained while a location client is connected to Location Service. Assuming that no other apps are connected to Location Services, if you disconnect the client and then sometime later call getLastLocation() (/reference/com/google/android/gms/location/LocationClient.html#getLastLocation()), the result may be out of date.

```
public class MainActivity extends FragmentActivity implements
        GooglePlayServicesClient.ConnectionCallbacks,
        GooglePlayServicesClient.OnConnectionFailedListener {
    ...
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        /*
         * Create a new location client, using the enclosing class to
         * handle callbacks.
         */
        mLocationClient = new LocationClient(this, this, this);
        ...
    }
    ...
    /*
     * Called when the Activity becomes visible.
     */
    @Override
    protected void onStart() {
        super.onStart();
        // Connect the client.
        mLocationClient.connect();
    }
    ...
    /*
     * Called when the Activity is no longer visible.
     */
    @Override
    protected void onStop() {
        // Disconnecting the client invalidates it.
        mLocationClient.disconnect();
        super.onStop();
    }
    ...
}
```

Get the Current Location

To get the current location, call getLastLocation() (/reference/com/google/android/gms/location/LocationClient.html#getLastLocation()). For example:

```
public class MainActivity extends FragmentActivity implements
        GooglePlayServicesClient.ConnectionCallbacks,
        GooglePlayServicesClient.OnConnectionFailedListener {
    ...
    // Global variable to hold the current location
    Location mCurrentLocation;
    ...
    mCurrentLocation = mLocationClient.getLastLocation();
    ...
}
```

The next lesson, Receiving Location Updates (receive-location-updates.html), shows you how to receive periodic location updates from Location Services.