Request App Permissions

Every Android app runs in a limited-access sandbox. If an app needs to use resources or information outside of its own sandbox, the app has to request the appropriate *permission*. You declare that your app needs a permission by listing the permission in the app manifest (https://developer.android.com/guide/topics/manifest/manifest-intro.html) and then requesting that the user approve each permission at runtime (on Android 6.0 and higher).

This page describes how to use the Android Support Library (https://developer.android.com/tools/support-library/index.html) to check for and request permissions. The Android framework provides similar methods as of Android 6.0 (API level 23), but using the support library makes it easier to provide compatibility with older versions of Android.

Add permissions to the manifest

On all versions of Android, to declare that your app needs a permission, put a **`<uses-permission>`** (https://developer.android.com/guide/topics/manifest/uses-permission-element.html) element in your app manifest, as a child of the top-level **`<manifest>`** (https://developer.android.com/guide/topics/manifest/manifest-element.html) element. For example, an app that needs to send SMS messages would have this line in the manifest:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.example.snazzyapp">

    <uses-permission android:name="android.permission.SEND_SMS"/>
    <!-- other permissions go here -->

    <application ...>
        ...
    </application>
</manifest>
```

The system's behavior after you declare a permission depends on how sensitive the permission is. Some permissions are considered "normal" so the system immediately grants them upon installation. Other permissions are considered "dangerous" so the user must explicitly grant your app access. For more information about the different kinds of permissions, see Protection levels
(https://developer.android.com/guide/topics/permissions/overview.html#normal-dangerous).

Check for permissions

If your app needs a dangerous permission, you must check whether you have that permission every time you perform an operation that requires that permission. Beginning with Android 6.0 (API level 23), users can revoke permissions from any app at any time, even if the app targets a lower API level. So even if the app used the camera yesterday, it can't assume it still has that permission today.

To check if you have a permission, call the `ContextCompat.checkSelfPermission()`
(https://developer.android.com/reference/android/support/v4/content/ContextCompat.html#checkSelfPermission(android.content.Context, java.lang.String))
method. For example, this snippet shows how to check if the activity has permission to write to the calendar:

KOTLIN JAVA

```
if (ContextCompat.checkSelfPermission(thisActivity, Manifest.permissior
        != PackageManager.PERMISSION_GRANTED) {
    // Permission is not granted
}
```

If the app has the permission, the method returns `PERMISSION_GRANTED`
(https://developer.android.com/reference/android/content/pm/PackageManager.html#PERMISSION_GRANTED)
, and the app can proceed with the operation. If the app does not have the permission, the method returns `PERMISSION_DENIED`
(https://developer.android.com/reference/android/content/pm/PackageManager.html#PERMISSION_DENIED)
, and the app has to explicitly ask the user for permission.

Request permissions

When your app receives `PERMISSION_DENIED` (https://developer.android.com/reference/android/content/pm/PackageManager.html#PERMISSION_DENIED) from `checkSelfPermission()` (https://developer.android.com/reference/android/support/v4/content/ContextCompat.html#checkSelfPermission(android.content.Context, java.lang.String)), you need to prompt the user for that permission. Android provides several methods you can use to request a permission, such as `requestPermissions()` (https://developer.android.com/reference/android/support/v4/app/ActivityCompat.html#requestPermissions(android.app.Activity, java.lang.String[], int)), as shown in the code snippet below. Calling these methods brings up a standard Android dialog, which you cannot customize.

How this is displayed to the user depends on the device Android version as well as the target version of your application, as described in the Permissions Overview (https://developer.android.com/guide/topics/permissions/overview.html).

Explain why the app needs permissions

In some circumstances, you want to help the user understand why your app needs a permission. For example, if a user launches a photography app, the user probably won't be surprised that the app asks for permission to use the camera, but the user might not understand why the app wants access to the user's location or contacts. Before your app requests a permission, you should consider providing an explanation to the user. Keep in mind that you don't want to overwhelm the user with explanations; if you provide too many explanations, the user might find the app frustrating and remove it.

One approach you might use is to provide an explanation only if the user has already denied that permission request. Android provides a utility method, `shouldShowRequestPermissionRationale()` (https://developer.android.com/reference/android/support/v4/app/ActivityCompat.html#shouldShowRequestPermissionRationale(android.app.Activity, java.lang.String)), that returns `true` if the user has previously denied the request, and returns `false` if a user has denied a permission and selected the **Don't ask again** option in the permission request dialog, or if a device policy prohibits the permission.

If a user keeps trying to use functionality that requires a permission, but keeps denying the permission request, that probably means the user doesn't understand why the app needs the permission to provide that functionality. In a situation like that, it's probably a good idea to show an explanation.

More advice about how to create a good user experience when asking for permission is provided in App Permissions Best Practices (https://developer.android.com/training/permissions/usage-notes.html).

Request the permissions you need

If your app doesn't already have the permission it needs, the app must call one of the **`requestPermissions()`** (https://developer.android.com/reference/android/support/v4/app/ActivityCompat.html#requestPermissions(android.app.Activity, java.lang.String[], int)) methods to request the appropriate permissions. Your app passes the permissions it wants and an integer *request code* that you specify to identify this permission request. This method functions asynchronously. It returns right away, and after the user responds to the prompt, the system calls the app's callback method with the results, passing the same request code that the app passed to **`requestPermissions()`** (https://developer.android.com/reference/android/support/v4/app/ActivityCompat.html#requestPermissions(android.app.Activity, java.lang.String[], int))
.

The following code checks if the app has permission to read the user's contacts. If it does not have permission it checks if it should show an explanation for needing the permission, and if no explanation is needed, it requests the permission:

KOTLIN JAVA

```
// Here, thisActivity is the current activity
if (ContextCompat.checkSelfPermission(thisActivity,
        Manifest.permission.READ_CONTACTS)
        != PackageManager.PERMISSION_GRANTED) {

    // Permission is not granted
    // Should we show an explanation?
    if (ActivityCompat.shouldShowRequestPermissionRationale(thisActivity,
            Manifest.permission.READ_CONTACTS)) {
        // Show an explanation to the user *asynchronously* -- don't block
        // this thread waiting for the user's response! After the user
        // sees the explanation, try again to request the permission.
    } else {
        // No explanation needed; request the permission
        ActivityCompat.requestPermissions(thisActivity,
                new String[]{Manifest.permission.READ_CONTACTS},
                MY_PERMISSIONS_REQUEST_READ_CONTACTS);

        // MY_PERMISSIONS_REQUEST_READ_CONTACTS is an
        // app-defined int constant. The callback method gets the
        // result of the request.
    }
} else {
    // Permission has already been granted
}
```

The prompt shown by the system describes the permission group (https://developer.android.com/guide/topics/permissions/overview.html#perm-groups) your app needs access to, not the specific permission.

Note: When your app calls `requestPermissions()` (https://developer.android.com/reference/android/support/v4/app/ActivityCompat.html#requestPermissions(android.app.Activity, java.lang.String[], int)), the system shows a standard dialog box to the user. Your app *cannot* configure or alter that dialog box. If you need to provide any information or explanation to the user, you should do that before you call `requestPermissions()` (https://developer.android.com/reference/android/support/v4/app/ActivityCompat.html#requestPermissions(android.app.Activity, java.lang.String[], int)), as described in Explain why the app needs permissions (#explain).

Handle the permissions request response

When the user responds to your app's permission request, the system invokes your app's `onRequestPermissionsResult()` (https://developer.android.com/reference/android/support/v4/app/ActivityCompat.OnRequestPermissionsResultCallback.html#onRequestPermissionsResult(int, java.lang.String[], int[])) method, passing it the user response. Your app has to override that method to find out whether the permission was granted. The callback is passed the same request code you passed to `requestPermissions()` (https://developer.android.com/reference/android/support/v4/app/ActivityCompat.html#requestPermissions(android.app.Activity, java.lang.String[], int)). For example, if an app requests `READ_CONTACTS` (https://developer.android.com/reference/android/Manifest.permission.html#READ_CONTACTS) access it might have the following callback method:

KOTLIN JAVA

```
@Override
public void onRequestPermissionsResult(int requestCode,
        String permissions[], int[] grantResults) {
    switch (requestCode) {
        case MY_PERMISSIONS_REQUEST_READ_CONTACTS: {
            // If request is cancelled, the result arrays are empty.
            if (grantResults.length > 0
                && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                // permission was granted, yay! Do the
                // contacts-related task you need to do.
            } else {
                // permission denied, boo! Disable the
                // functionality that depends on this permission.
            }
            return;
        }

        // other 'case' lines to check for other
        // permissions this app might request.
    }
}
```

The dialog box shown by the system describes the permission group (https://developer.android.com/guide/topics/security/permissions.html#perm-groups) your app needs access to; it does not list the specific permission. For example, if you request the `READ_CONTACTS` (https://developer.android.com/reference/android/Manifest.permission.html#READ_CONTACTS) permission, the system dialog box just says your app needs access to the device's contacts. The user only needs to grant permission once for each permission group. If your app requests any other permissions in that group (that are listed in your app manifest), the system automatically grants them. When you request the permission, the system calls your `onRequestPermissionsResult()` (https://developer.android.com/reference/android/support/v4/app/ActivityCompat.OnRequestPermissionsResultCallback.html#onRequestPermissionsResult(int, java.lang.String[], int[])) callback method and passes `PERMISSION_GRANTED` (https://developer.android.com/reference/android/content/pm/PackageManager.html#PERMISSION_GRANTED) , the same way it would if the user had explicitly granted your request through the system dialog box.

Note: Your app still needs to explicitly request every permission it needs, even if the user has already granted another permission in the same group. In addition, the grouping of permissions into groups may change in future Android releases. Your code should not rely on the assumption that particular permissions are or are not in the same group.

For example, suppose you list both `READ_CONTACTS` (https://developer.android.com/reference/android/Manifest.permission.html#READ_CONTACTS) and

`WRITE_CONTACTS` (https://developer.android.com/reference/android/Manifest.permission.html#WRITE_CONTACTS) in your app manifest. If you request `READ_CONTACTS` (https://developer.android.com/reference/android/Manifest.permission.html#READ_CONTACTS) and the user grants the permission, and you then request `WRITE_CONTACTS` (https://developer.android.com/reference/android/Manifest.permission.html#WRITE_CONTACTS), the system immediately grants you that permission without interacting with the user.

If the user denies a permission request, your app should take appropriate action. For example, your app might show a dialog explaining why it could not perform the user's requested action that needs that permission.

When the system asks the user to grant a permission, the user has the option of telling the system not to ask for that permission again. In that case, any time an app uses `requestPermissions()` (https://developer.android.com/reference/android/support/v4/app/ActivityCompat.html#requestPermissions(android.app.Activity, java.lang.String[], int)) to ask for that permission again, the system immediately denies the request. The system calls your `onRequestPermissionsResult()` (https://developer.android.com/reference/android/support/v4/app/ActivityCompat.OnRequestPermissionsResultCallback.html#onRequestPermissionsResult(int, java.lang.String[], int[])) callback method and passes `PERMISSION_DENIED` (https://developer.android.com/reference/android/content/pm/PackageManager.html#PERMISSION_DENIED) , the same way it would if the user had explicitly rejected your request again. The method also returns `false` if a device policy prohibits the app from having that permission. This means that when you call `requestPermissions()` (https://developer.android.com/reference/android/support/v4/app/ActivityCompat.html#requestPermissions(android.app.Activity, java.lang.String[], int)) , you cannot assume that any direct interaction with the user has taken place.

To provide the best user experience when asking for app permissions, also see App Permissions Best Practices (https://developer.android.com/training/permissions/usage-notes.html).

Additional resources

For additional information about permissions, read these articles:

- Permissions overview (https://developer.android.com/guide/topics/permissions/overview)
- App permissions best practices (https://developer.android.com/training/permissions/usage-notes)

To learn more about requesting permissions, download the following sample apps:

- Android RuntimePermissionsBasic Sample (https://github.com/googlesamples/android-RuntimePermissionsBasic/)
- Android RuntimePermissions Sample (https://github.com/googlesamples/android-RuntimePermissions/).

Previous
← Overview (https://developer.android.com/guide/topics/permissions/overview)

Next
App permissions best practices → (https://developer.android.com/training/permissions/usage-notes)