The Manifest File

Before the Android system can start an app component, the system must know that the component exists by reading the app's AndroidManifest.xml file (the "manifest" file). Your app must declare all its components in this file, which must be at the root of the app project directory.

The manifest does a number of things in addition to declaring the app's components, such as:

* Identify any user permissions the app requires, such as Internet access or read-access to the user's contacts.
* Declare the minimum [API Level](https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels) required by the app, based on which APIs the app uses.
* Declare hardware and software features used or required by the app, such as a camera, bluetooth services, or a multitouch screen.
* API libraries the app needs to be linked against (other than the Android framework APIs), such as the [Google Maps library](http://code.google.com/android/add-ons/google-apis/maps-overview.html).
* And more

Declaring components

The primary task of the manifest is to inform the system about the app's components. For example, a manifest file can declare an activity as follows:

<?xml version="1.0" encoding="utf-8"?>  
<manifest ... >  
    <application android:icon="@drawable/app\_icon.png" ... >  
        <activity android:name="com.example.project.ExampleActivity"  
                  android:label="@string/example\_label" ... >  
        </activity>  
        ...  
    </application>  
</manifest>

In the [<application>](https://developer.android.com/guide/topics/manifest/application-element.html) element, the android:icon attribute points to resources for an icon that identifies the app.

In the [<activity>](https://developer.android.com/guide/topics/manifest/activity-element.html) element, the android:name attribute specifies the fully qualified class name of the [Activity](https://developer.android.com/reference/android/app/Activity.html) subclass and the android:labelattribute specifies a string to use as the user-visible label for the activity.

You must declare all app components this way:

* [<activity>](https://developer.android.com/guide/topics/manifest/activity-element.html) elements for activities
* [<service>](https://developer.android.com/guide/topics/manifest/service-element.html) elements for services
* [<receiver>](https://developer.android.com/guide/topics/manifest/receiver-element.html) elements for broadcast receivers
* [<provider>](https://developer.android.com/guide/topics/manifest/provider-element.html) elements for content providers

**Activities, services,** and **content** **providers** that you include in your source but do not declare in the manifest are not visible to the system and, consequently, can never run. However, broadcast receivers can be either declared in the manifest or created dynamically in code (as [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html) objects) and registered with the system by calling [registerReceiver()](https://developer.android.com/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter)).

For more about how to structure the manifest file for your app, see [The AndroidManifest.xml File](https://developer.android.com/guide/topics/manifest/manifest-intro.html) documentation.

Declaring component capabilities

As discussed above, in [Activating Components](https://developer.android.com/guide/components/fundamentals.html#ActivatingComponents), you can use an [Intent](https://developer.android.com/reference/android/content/Intent.html) to start activities, services, and broadcast receivers. You can do so by explicitly naming the target component (using the component class name) in the intent. However, the real power of intents lies in the concept of *implicit intents*. An implicit intent simply describes the type of action to perform (and, optionally, the data upon which you’d like to perform the action) and allows the system to find a component on the device that can perform the action and start it. If there are multiple components that can perform the action described by the intent, then the user selects which one to use.

The way the system identifies the components that can respond to an intent is by comparing the intent received to the *intent filters* provided in the manifest file of other apps on the device.

When you declare an activity in your app's manifest, you can optionally include intent filters that declare the capabilities of the activity so it can respond to intents from other apps. You can declare an intent filter for your component by adding an [<intent-filter>](https://developer.android.com/guide/topics/manifest/intent-filter-element.html) element as a child of the component's declaration element.

For example, if you've built an email app with an activity for composing a new email, you can declare an intent filter to respond to "send" intents (in order to send a new email) like this:

<manifest ... >  
    ...  
    <application ... >  
        <activity android:name="com.example.project.ComposeEmailActivity">  
            <intent-filter>  
                <action android:name="android.intent.action.SEND" />  
                <data android:type="\*/\*" />  
                <category android:name="android.intent.category.DEFAULT" />  
            </intent-filter>  
        </activity>  
    </application>  
</manifest>

Then, if another app creates an intent with the [ACTION\_SEND](https://developer.android.com/reference/android/content/Intent.html#ACTION_SEND) action and passes it to [startActivity()](https://developer.android.com/reference/android/app/Activity.html#startActivity(android.content.Intent)), the system may start your activity so the user can draft and send an email.

For more about creating intent filters, see the [Intents and Intent Filters](https://developer.android.com/guide/components/intents-filters.html) document.

Declaring app requirements

There are a variety of devices powered by Android and not all of them provide the same features and capabilities. In order to prevent your app from being installed on devices that lack features needed by your app, it's important that you clearly define a profile for the types of devices your app supports by declaring device and software requirements in your manifest file. Most of these declarations are informational only and the system does not read them, but external services such as Google Play do read them in order to provide filtering for users when they search for apps from their device.

For example, if your app requires a camera and uses APIs introduced in Android 2.1 ([API Level](https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels) 7), you should declare these as requirements in your manifest file like this:

<manifest ... >  
    <uses-feature android:name="android.hardware.camera.any"  
                  android:required="true" />  
    <uses-sdk android:minSdkVersion="7" android:targetSdkVersion="19" />  
    ...  
</manifest>

Now, devices that do *not* have a camera and have an Android version *lower* than 2.1 cannot install your app from Google Play.

However, you can also declare that your app uses the camera, but does not *require* it. In that case, your app must set the [required](https://developer.android.com/guide/topics/manifest/uses-feature-element.html#required) attribute to "false"and check at runtime whether the device has a camera and disable any camera features as appropriate.

More information about how you can manage your app's compatibility with different devices is provided in the [Device Compatibility](https://developer.android.com/guide/practices/compatibility.html) document.