

Android Developers

App Manifest

Every application must have an `AndroidManifest.xml` file (with precisely that name) in its root directory. The manifest file provides essential information about your app to the Android system, which the system must have before it can run any of the app's code.

Among other things, the manifest file does the following:

- It names the Java package for the application. The package name serves as a unique identifier for the application.
- It describes the components of the application, which include the activities, services, broadcast receivers, and content providers that compose the application. It also names the classes that implement each of the components and publishes their capabilities, such as the `Intent` (<https://developer.android.com/reference/android/content/Intent.html>) messages that they can handle. These declarations inform the Android system of the components and the conditions in which they can be launched.
- It determines the processes that host the application components.
- It declares the permissions that the application must have in order to access protected parts of the API and interact with other applications. It also declares the permissions that others are required to have in order to interact with the application's components.
- It lists the `Instrumentation` (<https://developer.android.com/reference/android/app/Instrumentation.html>) classes that provide profiling and other information as the application runs. These declarations are present in the manifest only while the application is being developed and are removed before the application is published.
- It declares the minimum level of the Android API that the application requires.
- It lists the libraries that the application must be linked against.

Note: As you prepare your Android app to run on Chromebooks, there are some important hardware and software feature limitations that you should consider. See the App Manifest Compatibility for Chromebooks (<https://developer.android.com/topic/arc/manifest.html>) document for more information.

In this document

Manifest file structure

File conventions

File features

Intent filters

Icons and labels

Permissions

Libraries

Manifest file structure

The code snippet below shows the general structure of the manifest file and every element that it can contain. Each element, along with all of its attributes, is fully documented in a separate file.

Tip: To view detailed information about any of the elements that are mentioned within the text of this document, simply click the element name.

Here is an example of the manifest file:

```
<?xml version="1.0" encoding="utf-8"?>

<manifest> (https://developer.android.com/guide/topics/manifest/manifest-element.html)

    <uses-permission /> (https://developer.android.com/guide/topics/manifest/uses-permission-element.html)
    <permission /> (https://developer.android.com/guide/topics/manifest/permission-element.html)
    <permission-tree /> (https://developer.android.com/guide/topics/manifest/permission-tree-element.html)
    <permission-group /> (https://developer.android.com/guide/topics/manifest/permission-group-element.html)
    <instrumentation /> (https://developer.android.com/guide/topics/manifest/instrumentation-element.html)
    <uses-sdk /> (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html)
    <uses-configuration /> (https://developer.android.com/guide/topics/manifest/uses-configuration-element.html)
    <uses-feature /> (https://developer.android.com/guide/topics/manifest/uses-feature-element.html)
    <supports-screens /> (https://developer.android.com/guide/topics/manifest/supports-screens-element.html)
    <compatible-screens /> (https://developer.android.com/guide/topics/manifest/compatible-screens-element.html)
    <supports-gl-texture /> (https://developer.android.com/guide/topics/manifest/supports-gl-texture-element.html)

    <application> (https://developer.android.com/guide/topics/manifest/application-element.html)

        <activity> (https://developer.android.com/guide/topics/manifest/activity-element.html)
            <intent-filter> (https://developer.android.com/guide/topics/manifest/intent-filter-element.html)
                <action /> (https://developer.android.com/guide/topics/manifest/action-element.html)
                <category /> (https://developer.android.com/guide/topics/manifest/category-element.html)
                <data /> (https://developer.android.com/guide/topics/manifest/data-element.html)
            </intent-filter> (https://developer.android.com/guide/topics/manifest/intent-filter-element.html)
            <meta-data /> (https://developer.android.com/guide/topics/manifest/meta-data-element.html)
        </activity> (https://developer.android.com/guide/topics/manifest/activity-element.html)

        <activity-alias> (https://developer.android.com/guide/topics/manifest/activity-alias-element.html)
            <intent-filter> (https://developer.android.com/guide/topics/manifest/intent-filter-element.html)
            <meta-data /> (https://developer.android.com/guide/topics/manifest/meta-data-element.html)
        </activity-alias> (https://developer.android.com/guide/topics/manifest/activity-alias-element.html)

        <service> (https://developer.android.com/guide/topics/manifest/service-element.html)
```

Take a one-minute

```

        <intent-filter> (https://developer.android.com/guide/topics/manifest/intent-filter-element.html)
        <meta-data/> (https://developer.android.com/guide/topics/manifest/meta-data-element.html)
    </service> (https://developer.android.com/guide/topics/manifest/service-element.html)

    <receiver> (https://developer.android.com/guide/topics/manifest/receiver-element.html)
        <intent-filter> (https://developer.android.com/guide/topics/manifest/intent-filter-element.html)
        <meta-data /> (https://developer.android.com/guide/topics/manifest/meta-data-element.html)
    </receiver> (https://developer.android.com/guide/topics/manifest/receiver-element.html)

    <provider> (https://developer.android.com/guide/topics/manifest/provider-element.html)
        <grant-uri-permission /> (https://developer.android.com/guide/topics/manifest/grant-uri-permission-element.html)
        <meta-data /> (https://developer.android.com/guide/topics/manifest/meta-data-element.html)
        <path-permission /> (https://developer.android.com/guide/topics/manifest/path-permission-element.html)
    </provider> (https://developer.android.com/guide/topics/manifest/provider-element.html)

    <uses-library /> (https://developer.android.com/guide/topics/manifest/uses-library-element.html)

</application> (https://developer.android.com/guide/topics/manifest/application-element.html)

</manifest> (https://developer.android.com/guide/topics/manifest/manifest-element.html)

```

The following list contains all of the elements that can appear in the manifest file, in alphabetical order:

- `<action>` (<https://developer.android.com/guide/topics/manifest/action-element.html>)
- `<activity>` (<https://developer.android.com/guide/topics/manifest/activity-element.html>)
- `<activity-alias>` (<https://developer.android.com/guide/topics/manifest/activity-alias-element.html>)
- `<application>` (<https://developer.android.com/guide/topics/manifest/application-element.html>)
- `<category>` (<https://developer.android.com/guide/topics/manifest/category-element.html>)
- `<data>` (<https://developer.android.com/guide/topics/manifest/data-element.html>)
- `<grant-uri-permission>` (<https://developer.android.com/guide/topics/manifest/grant-uri-permission-element.html>)
- `<instrumentation>` (<https://developer.android.com/guide/topics/manifest/instrumentation-element.html>)
- `<intent-filter>` (<https://developer.android.com/guide/topics/manifest/intent-filter-element.html>)
- `<manifest>` (<https://developer.android.com/guide/topics/manifest/manifest-element.html>)
- `<meta-data>` (<https://developer.android.com/guide/topics/manifest/meta-data-element.html>)

Take a one-minute

- `<permission>` (<https://developer.android.com/guide/topics/manifest/permission-element.html>)
- `<permission-group>` (<https://developer.android.com/guide/topics/manifest/permission-group-element.html>)
- `<permission-tree>` (<https://developer.android.com/guide/topics/manifest/permission-tree-element.html>)
- `<provider>` (<https://developer.android.com/guide/topics/manifest/provider-element.html>)
- `<receiver>` (<https://developer.android.com/guide/topics/manifest/receiver-element.html>)
- `<service>` (<https://developer.android.com/guide/topics/manifest/service-element.html>)
- `<supports-screens>` (<https://developer.android.com/guide/topics/manifest/supports-screens-element.html>)
- `<uses-configuration>` (<https://developer.android.com/guide/topics/manifest/uses-configuration-element.html>)
- `<uses-feature>` (<https://developer.android.com/guide/topics/manifest/uses-feature-element.html>)
- `<uses-library>` (<https://developer.android.com/guide/topics/manifest/uses-library-element.html>)
- `<uses-permission>` (<https://developer.android.com/guide/topics/manifest/uses-permission-element.html>)
- `<uses-sdk>` (<https://developer.android.com/guide/topics/manifest/uses-sdk-element.html>)

Note: These are the only legal elements – you cannot add your own elements or attributes.

File conventions

This section describes the conventions and rules that apply generally to all of the elements and attributes in the manifest file.

Elements

Only the `<manifest>` (<https://developer.android.com/guide/topics/manifest/manifest-element.html>) and `<application>` (<https://developer.android.com/guide/topics/manifest/application-element.html>) elements are required. They each must be present and can occur only once. Most of the other elements can occur many times or not at all. However, at least some of them must be useful.

Take a one-minute

If an element contains anything at all, it contains other elements. All of the values are set through attributes, not as character data within an element.

Elements at the same level are generally not ordered. For example, the `<activity>` (<https://developer.android.com/guide/topics/manifest/activity-element.html>), `<provider>` (<https://developer.android.com/guide/topics/manifest/provider-element.html>), and `<service>` (<https://developer.android.com/guide/topics/manifest/service-element.html>) elements can be intermixed in any sequence. There are two key exceptions to this rule:

- An `<activity-alias>` (<https://developer.android.com/guide/topics/manifest/activity-alias-element.html>) element must follow the `<activity>` (<https://developer.android.com/guide/topics/manifest/activity-element.html>) for which it is an alias.
- The `<application>` (<https://developer.android.com/guide/topics/manifest/application-element.html>) element must be the last element inside the `<manifest>` (<https://developer.android.com/guide/topics/manifest/manifest-element.html>) element. In other words, the `</application>` closing tag must appear immediately before the `</manifest>` closing tag.

Attributes

In a formal sense, all attributes are optional. However, there are some attributes that must be specified so that an element can accomplish its purpose. Use the documentation as a guide. For truly optional attributes, it mentions a default value or states what happens in the absence of a specification.

Except for some attributes of the root `<manifest>` (<https://developer.android.com/guide/topics/manifest/manifest-element.html>) element, all attribute names begin with an `android:` prefix. For example, `android:alwaysRetainTaskState`. Because the prefix is universal, the documentation generally omits it when referring to attributes by name.

Declaring class names

Many elements correspond to Java objects, including elements for the application itself (the `<application>` (<https://developer.android.com/guide/topics/manifest/application-element.html>) element) and its principal components: activities (`<activity>` (<https://developer.android.com/guide/topics/manifest/activity-element.html>)), services (`<service>` (<https://developer.android.com/guide/topics/manifest/service-element.html>)), broadcast receivers (`<receiver>` (<https://developer.android.com/guide/topics/manifest/receiver-element.html>)), and content providers (`<provider>` (<https://developer.android.com/guide/topics/manifest/provider-element.html>)).

If you define a subclass of one of these component classes (`Activity`, `Service`, `BroadcastReceiver`, or `ContentProvider`), you must declare the class name in the `android:name` attribute of the corresponding element.

(<https://developer.android.com/reference/android/app/Activity.html>), **Service** (<https://developer.android.com/reference/android/app/Service.html>), **BroadcastReceiver** (<https://developer.android.com/reference/android/content/BroadcastReceiver.html>), and **ContentProvider** (<https://developer.android.com/reference/android/content/ContentProvider.html>)), the subclass is declared through a **name** attribute. The name must include the full package designation. For example, a **Service** (<https://developer.android.com/reference/android/app/Service.html>) subclass might be declared as follows:

```
<manifest . . . >
    <application . . . >
        <service android:name="com.example.project.SecretService" . . . >
            . . .
        </service>
        . . .
    </application>
</manifest>
```

However, if the first character of the string is a period, the application's package name (as specified by the **<manifest>** (<https://developer.android.com/guide/topics/manifest/manifest-element.html>) element's **package** (<https://developer.android.com/guide/topics/manifest/manifest-element.html#package>) attribute) is appended to the string. The following assignment is the same as that shown above:

```
<manifest package="com.example.project" . . . >
    <application . . . >
        <service android:name=".SecretService" . . . >
            . . .
        </service>
        . . .
    </application>
</manifest>
```

When starting a component, the Android system creates an instance of the named subclass. If a subclass isn't specified, it creates an instance of the base class.

Multiple values

If more than one value can be specified, the element is almost always repeated, rather than multiple values being listed within a single element. For example, an intent filter can list several actions:

```
<intent-filter . . . >
    <action android:name="android.intent.action.MAIN" />
    <action android:name="android.intent.action.VIEW" />
    <action android:name="android.intent.action.SEARCH" />
</intent-filter>
```

Take a one-minute

```

        <action android:name="android.intent.action.INSERT" />
        <action android:name="android.intent.action.DELETE" />
        . . .
    </intent-filter>

```

Resource values

Some attributes have values that can be displayed to users, such as a label and an icon for an activity. The values of these attributes should be localized and set from a resource or theme. Resource values are expressed in the following format:

```
@[<i>package</i>:]<i>type</i>/<i>name</i>
```

You can omit the *package* name if the resource is in the same package as the application. The *type* is a type of resource, such as *string* or *drawable*, and the *name* is the name that identifies the specific resource. Here is an example:

```
<activity android:icon="@drawable/smallPic" . . . >
```

The values from a theme are expressed similarly, but with an initial `?` instead of `@`:

```
?[<i>package</i>:]<i>type</i>/<i>name</i>
```

String values

Where an attribute value is a string, you must use double backslashes (`\\`) to escape characters, such as `\\n` for a newline or `\\uxxxx` for a Unicode character.

File features

The following sections describe the way that some Android features are reflected in the manifest file.

Intent filters

The core components of an application, such as its activities, services, and broadcast receivers, are activated by *intents*. An intent is a bundle of information (an `Intent` (<https://developer.android.com/reference/android/content/Intent.html>) object) describing a desired action, including the data to be acted upon, the category of component that should perform the action, and other pertinent instructions. The Android system can respond to the intent, and it launches a new instance of the component that is the `Intent`.

Take a one-minute

(<https://developer.android.com/reference/android/content/Intent.html>) object.

The components advertise the types of intents that they can respond to through *intent filters*. Since the Android system must learn the intents that a component can handle before it launches the component, intent filters are specified in the manifest as `<intent-filter>`

(<https://developer.android.com/guide/topics/manifest/intent-filter-element.html>) elements. A component can have any number of filters, each one describing a different capability.

An intent that explicitly names a target component activates that component, so the filter doesn't play a role. An intent that doesn't specify a target by name can activate a component only if it can pass through one of the component's filters.

For information about how `Intent` (<https://developer.android.com/reference/android/content/Intent.html>) objects are tested against intent filters, see the [Intents and Intent Filters](https://developer.android.com/guide/components/intents-filters.html) (<https://developer.android.com/guide/components/intents-filters.html>) document.

Icons and labels

A number of elements have `icon` and `label` attributes for a small icon and a text label that can be displayed to users. Some also have a `description` attribute for longer, explanatory text that can also be shown on-screen. For example, the `<permission>` (<https://developer.android.com/guide/topics/manifest/permission-element.html>) element has all three of these attributes so that when the user is asked whether to grant the permission to an application that has requested it, an icon representing the permission, the name of the permission, and a description of what it entails are all presented to the user.

In every case, the icon and label that are set in a containing element become the default `icon` and `label` settings for all of the container's subelements. Thus, the icon and label that are set in the `<application>` (<https://developer.android.com/guide/topics/manifest/application-element.html>) element are the default icon and label for each of the application's components. Similarly, the icon and label that are set for a component, such as an `<activity>` (<https://developer.android.com/guide/topics/manifest/activity-element.html>) element, are the default settings for each of the component's `<intent-filter>` (<https://developer.android.com/guide/topics/manifest/intent-filter-element.html>) elements. If an `<application>` (<https://developer.android.com/guide/topics/manifest/application-element.html>) element sets a label, but an activity and its intent filter do not, the application label is treated as the label for both the activity and the intent filter.

The icon and label that are set for an intent filter represent a component whenever the component is presented to the user and fulfills the function that is advertised by the filter. For example, a filter with `android.intent.action.MAIN` and `android.intent.category.LAUNCHER` settings advertises an activity as one that should be displayed in the application launcher. The icon and label that are set for the filter are the icon and label that should be displayed in the launcher.

Take a one-minute

Permissions

A *permission* is a restriction that limits access to a part of the code or to data on the device. The limitation is imposed to protect critical data and code that could be misused to distort or damage the user experience.

Each permission is identified by a unique label. Often the label indicates the action that's restricted. Here are some permissions that are defined by Android:

- `android.permission.CALL_EMERGENCY_NUMBERS`
- `android.permission.READ_OWNER_DATA`
- `android.permission.SET_WALLPAPER`
- `android.permission.DEVICE_POWER`

A feature can be protected by only one permission.

If an application needs access to a feature that is protected by a permission, it must declare that it requires the permission with a `<uses-permission>` (<https://developer.android.com/guide/topics/manifest/uses-permission-element.html>) element in the manifest. When the application is installed on the device, the installer determines whether to grant the requested permission by checking the authorities that signed the application's certificates and, in some cases, asking the user. If the permission is granted, the application is able to use the protected features. If not, its attempts to access those features fail without any notification to the user.

An application can also protect its own components with permissions. It can employ any of the permissions that are defined by Android, as listed in `android.Manifest.permission` (<https://developer.android.com/reference/android/Manifest.permission.html>), or declared by other applications. It can also define its own. A new permission is declared with the `<permission>` (<https://developer.android.com/guide/topics/manifest/permission-element.html>) element. For example, an activity could be protected as follows:

```
<manifest . . . >
    <permission android:name="com.example.project.DEBIT_ACCT" . . . />
    <uses-permission android:name="com.example.project.DEBIT_ACCT" />
    . . .
    <application . . . >
        <activity android:name="com.example.project.FreneticActivity"
            android:permission="com.example.project.DEBIT_ACCT"
            . . . />
```

Take a one-minute

```

        . . . . >
        . . . .
    </activity>
</application>
</manifest>

```

Note that, in this example, the `DEBIT_ACCT` permission is not only declared with the `<permission>` (<https://developer.android.com/guide/topics/manifest/permission-element.html>) element, its use is also requested with the `<uses-permission>` (<https://developer.android.com/guide/topics/manifest/uses-permission-element.html>) element. You must request its use in order for other components of the application to launch the protected activity, even though the protection is imposed by the application itself.

If, in the same example shown above, the `permission` attribute was set to a permission that is declared elsewhere, such as `android.permission.CALL_EMERGENCY_NUMBERS`, it would not be necessary to declare it again with a `<permission>` (<https://developer.android.com/guide/topics/manifest/permission-element.html>) element. However, it would still be necessary to request its use with `<uses-permission>` (<https://developer.android.com/guide/topics/manifest/uses-permission-element.html>).

The `<permission-tree>` (<https://developer.android.com/guide/topics/manifest/permission-tree-element.html>) element declares a namespace for a group of permissions that are defined in code, and the `<permission-group>` (<https://developer.android.com/guide/topics/manifest/permission-group-element.html>) defines a label for a set of permissions, both those declared in the manifest with `<permission>` (<https://developer.android.com/guide/topics/manifest/permission-element.html>) elements and those declared elsewhere. This affects only how the permissions are grouped when presented to the user. The `<permission-group>` (<https://developer.android.com/guide/topics/manifest/permission-group-element.html>) element does not specify the permissions that belong to the group, but it gives the group a name. You can place a permission in the group by assigning the group name to the `<permission>` (<https://developer.android.com/guide/topics/manifest/permission-element.html>) element's `permissionGroup` (<https://developer.android.com/guide/topics/manifest/permission-element.html#pgroup>) attribute.

Libraries

Every application is linked against the default Android library, which includes the basic packages for building applications (with common classes such as Activity, Service, Intent, View, Button, Application, and ContentProvider).

However, some packages reside in their own libraries. If your application uses code from any of these packages, it must explicitly declare them. The manifest must contain a separate `<uses-library>` (<https://developer.android.com/guide/topics/manifest/uses-library-element.html>) element for each library. Take a one-minute tour of the `<uses-library>` element (<https://developer.android.com/guide/topics/manifest/uses-library-element.html>).

`element.html`) element to name each of the libraries. You can find the library name in the documentation for the package.

Take a one-minute