Distributing to Specific Screens

In this document

Although we recommend that you design your application to function properly on multiple configurations of screen size and density, you can instead choose to limit the distribution of your application to certain types of screens, such as only tablets and other large devices or only handsets and similar-sized devices. To do so, you can enable filtering by external services such as Google Play by adding elements to your manifest file that specify the screen configurations your application supports.

However, before you decide to restrict your application to certain screen configurations, you should understand the techniques for supporting multiple screens and implement them to the best of your ability. By supporting multiple screens, your application can be made available to the greatest number of users with different devices, using a single APK.

Declaring an App is Only for Handsets

Because the system generally scales applications to fit larger screens well, you shouldn't need to filter your application from larger screens. As long as you follow the Best Practices for Screen Independence, your application should work well on larger screens such as tablets. However, you might discover that your application can't scale up well or perhaps you've decided to publish two versions of your application for different screen configurations. In such a case, you can use the `<compatible-screens>` element to manage the distribution of your application based on combinations of screen size and density. External services such as Google Play use this information to apply filtering to your application, so that only devices that have a screen configuration with which you declare compatibility can download your application.

The `<compatible-screens>` element must contain one or more `<screen>` elements. Each `<screen>` element specifies a screen configuration with which your application is compatible, using both the `android:screenSize` and `android:screenDensity` attributes. Each `<screen>` element **must include both attributes** to specify an individual screen configuration—if either attribute is missing, then the element is invalid (external services such as Google Play will ignore it).

For example, if your application is compatible with only small and normal size screens, regardless of screen density, you must specify eight different `<screen>` elements, because each screen size has four density configurations. You must declare each one of these; any combination of size and density that you do *not* specify is considered a screen configuration with which your application is *not* compatible. Here's what the manifest entry looks like if your application is compatible with only small and normal screen sizes:

```
<manifest ... >
    <compatible-screens>
        <!-- all small size screens -->
        <screen android:screenSize="small" android:screenDensity="ldpi" />
        <screen android:screenSize="small" android:screenDensity="mdpi" />
        <screen android:screenSize="small" android:screenDensity="hdpi" />
        <screen android:screenSize="small" android:screenDensity="xhdpi" />
        <!-- all normal size screens -->
        <screen android:screenSize="normal" android:screenDensity="ldpi" />
        <screen android:screenSize="normal" android:screenDensity="mdpi" />
        <screen android:screenSize="normal" android:screenDensity="hdpi" />
        <screen android:screenSize="normal" android:screenDensity="xhdpi" />
    </compatible-screens>
    ...
    <application ... >
        ...
    <application>
</manifest>
```

> **Note:** Although you can also use the `<compatible-screens>` element for the reverse scenario (when your application is not compatible with smaller screens), it's easier if you instead use the `<supports-screens>` as discussed in the next section, because it doesn't require you to specify each screen density your application supports.

Declaring an App is Only for Tablets

If you don't want your app to be used on handsets (perhaps your app truly makes sense only on a large screen) or you need time to optimize it for smaller screens, you can prevent small-screen devices from downloading your app by using the `<supports-screens>` manifest element.

For example, if you want your application to be available only to tablet devices, you can declare the element in your manifest like this:

```
<manifest ... >
    <supports-screens android:smallScreens="false"
                      android:normalScreens="false"
                      android:largeScreens="true"
                      android:xlargeScreens="true"
                      android:requiresSmallestWidthDp="600" />
    ...
    <application ... >
        ...
    </application>
</manifest>
```

This describes your app's screen-size support in two different ways:

- It declares that the app does *not* support the screen sizes "small" and "normal", which are traditionally not tablets.
- It declares that the app requires a screen size with a minimum usable area that is at least 600dp wide.

The first technique is for devices that are running Android 3.1 or older, because those devices declare their size based on generalized screen sizes. The `requiresSmallestWidthDp` attribute is for devices running Android 3.2 and newer, which includes the capability for apps to specify size requirements based on a minimum number of density-independent pixels available. In this example, the app declares a minimum width requirement of 600dp, which generally implies a 7"-or-greater screen.

Your size choice might be different, of course, based on how well your design works on different screen sizes; for example, if your design works well only on screens that are 9" or larger, you might require a minimum width of 720dp.

The catch is that you must compile your application against Android 3.2 or higher in order to use the `requiresSmallestWidthDp` attribute. Older versions don't understand this attribute and will raise a compile-time error. The safest thing to do is develop your app against the platform that matches the API level you've set for minSdkVersion. When you're making final preparations to build your release candidate, change the build target to Android 3.2 and add the `requiresSmallestWidthDp` attribute. Android versions older than 3.2 simply ignore that XML attribute, so there's no risk of a runtime failure.

For more information about why the "smallest width" screen size is important for supporting different screen sizes, read New Tools for Managing Screen Sizes.

> **Caution:** If you use the `<supports-screens>` element for the reverse scenario (when your application is not compatible with *larger* screens) and set the larger screen size attributes to `"false"`, then external services such as Google Play **do not** apply filtering. Your application will still be available to larger screens, but when it runs, it will not resize to fit the screen. Instead, the system will emulate a handset screen size (about 320dp x 480dp; see Screen Compatibility Mode for more information). If you want to prevent your application from being downloaded on larger screens, use `<compatible-screens>`, as discussed in the previous section about Declaring an App is Only for Handsets.

Remember, you should strive to make your application available to as many devices as possible by applying all necessary techniques for supporting multiple screens. You should use `<compatible-screens>` or `<supports-screens>` only when you cannot provide compatibility on all screen configurations or you have decided to provide different versions of your application for different sets of screen configurations.

Publishing Multiple APKs for Different Screens

Although we recommend that you publish one APK for your application, Google Play allows you to publish multiple APKs for the same application when each APK supports a different set of screen configurations (as declared in the manifest file). For example, if you want to publish both a handset version and a tablet version of your application, but you're unable to make the same APK work for both screen sizes, you can actually publish two APKs for the same application listing. Depending on each device's screen configuration, Google Play will deliver it the APK that you've declared to support that device's screen.

Beware, however, that publishing multiple APKs for the same application is considered an advanced feature and **most applications should publish only one APK that can support a wide range of device configurations**. Supporting multiple screen sizes, especially, is within reason using a single APK, as long as you follow the guide to Supporting Multiple Screens.

If you need more information about how to publish multiple APKs on Google Play, read Multiple APK Support.