App Install Location

Quickview

- You can allow your application to install on the device's external storage.
- Some types of applications should **not** allow installation on the external storage.
- Installing on the external storage is ideal for large applications that are not tightly integrated with the system (most commonly, games).

In this document

See also

- `<manifest>`

Beginning with API Level 8, you can allow your application to be installed on the external storage (for example, the device's SD card). This is an optional feature you can declare for your application with the `android:installLocation` manifest attribute. If you do *not* declare this attribute, your application will be installed on the internal storage only and it cannot be moved to the external storage.

To allow the system to install your application on the external storage, modify your manifest file to include the `android:installLocation` attribute in the `<manifest>` element, with a value of either "`preferExternal`" or "`auto`". For example:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:installLocation="preferExternal"
    ... >
```

If you declare "`preferExternal`", you request that your application be installed on the external storage, but the system does not guarantee that your application will be installed on the external storage. If the external storage is full, the system will install it on the internal storage. The user can also move your application between the two locations.

If you declare "`auto`", you indicate that your application may be installed on the external storage, but you don't have a preference of install location. The system will decide where to install your application based on several factors. The user can also move your application between the two locations.

When your application is installed on the external storage:

- There is no effect on the application performance so long as the external storage is mounted on the device.
- The `.apk` file is saved on the external storage, but all private user data, databases, optimized `.dex` files, and extracted native code are saved on the internal device memory.
- The unique container in which your application is stored is encrypted with a randomly generated key that can be decrypted only by the device that originally installed it. Thus, an application installed on an SD card works for only one device.
- The user can move your application to the internal storage through the system settings.

Warning: When the user enables USB mass storage to share files with a computer or unmounts the SD card via the system settings, the external storage is unmounted from the device and all applications running on the external storage are immediately killed.

Backward Compatibility

Backward Compatibility

The ability for your application to install on the external storage is a feature available only on devices running API Level 8 (Android 2.2) or greater. Existing applications that were built prior to API Level 8 will always install on the internal storage and cannot be moved to the external storage (even on devices with API Level 8). However, if your application is designed to support an API Level *lower than* 8, you can choose to support this feature for devices with API Level 8 or greater and still be compatible with devices using an API Level lower than 8.

To allow installation on external storage and remain compatible with versions lower than API Level 8:

1. Include the `android:installLocation` attribute with a value of "`auto`" or "`preferExternal`" in the `<manifest>` element.
2. Leave your `android:minSdkVersion` attribute as is (something *less than* "8") and be certain that your application code uses only APIs compatible with that level.
3. In order to compile your application, change your build target to API Level 8. This is necessary because older Android libraries don't understand the `android:installLocation` attribute and will not compile your application when it's present.

When your application is installed on a device with an API Level lower than 8, the `android:installLocation` attribute is ignored and the application is installed on the internal storage.

> **Caution:** Although XML markup such as this will be ignored by older platforms, you must be careful not to use programming APIs introduced in API Level 8 while your `minSdkVersion` is less than "8", unless you perform the work necessary to provide backward compatibility in your code.

Applications That Should NOT Install on External Storage

When the user enables USB mass storage to share files with their computer (or otherwise unmounts or removes the external storage), any application installed on the external storage and currently running is killed. The system effectively becomes unaware of the application until mass storage is disabled and the external storage is remounted on the device. Besides killing the application and making it unavailable to the user, this can break some types of applications in a more serious way. In order for your application to consistently behave as expected, you **should not** allow your application to be installed on the external storage if it uses any of the following features, due to the cited consequences when the external storage is unmounted:

Services

> Your running `Service` will be killed and will not be restarted when external storage is remounted. You can, however, register for the `ACTION_EXTERNAL_APPLICATIONS_AVAILABLE` broadcast Intent, which will notify your application when applications installed on external storage have become available to the system again. At which time, you can restart your Service.

Alarm Services

> Your alarms registered with `AlarmManager` will be cancelled. You must manually re-register any alarms when external storage is remounted.

Input Method Engines

> Your IME will be replaced by the default IME. When external storage is remounted, the user can open system settings to enable your IME again.

Live Wallpapers

> Your running Live Wallpaper will be replaced by the default Live Wallpaper. When external storage is remounted, the user can select your Live Wallpaper again.

App Widgets

> Your App Widget will be removed from the home screen. When external storage is remounted, your App Widget will *not* be available for the user to select until the system resets the home application (usually not until a system reboot).

Account Managers

Your accounts created with `AccountManager` will disappear until external storage is remounted.

Sync Adapters

Your `AbstractThreadedSyncAdapter` and all its sync functionality will not work until external storage is remounted.

Device Administrators

Your `DeviceAdminReceiver` and all its admin capabilities will be disabled, which can have unforeseeable consequences for the device functionality, which may persist after external storage is remounted.

Broadcast Receivers listening for "boot completed"

The system delivers the `ACTION_BOOT_COMPLETED` broadcast before the external storage is mounted to the device. If your application is installed on the external storage, it can never receive this broadcast.

If your application uses any of the features listed above, you **should not** allow your application to install on external storage. By default, the system *will not* allow your application to install on the external storage, so you don't need to worry about your existing applications. However, if you're certain that your application should never be installed on the external storage, then you should make this clear by declaring `android:installLocation` with a value of "`internalOnly`". Though this does not change the default behavior, it explicitly states that your application should only be installed on the internal storage and serves as a reminder to you and other developers that this decision has been made.

Applications That Should Install on External Storage

In simple terms, anything that does not use the features listed in the previous section are safe when installed on external storage. Large games are more commonly the types of applications that should allow installation on external storage, because games don't typically provide additional services when inactive. When external storage becomes unavailable and a game process is killed, there should be no visible effect when the storage becomes available again and the user restarts the game (assuming that the game properly saved its state during the normal Activity lifecycle).

If your application requires several megabytes for the APK file, you should carefully consider whether to enable the application to install on the external storage so that users can preserve space on their internal storage.