Manage all files on a storage device

The majority of apps that require shared storage access can follow the best practices for sharing media files (/training/data-storage/use-cases#share-media-all) and sharing non-media files (/training/data-storage/use-cases#sharing-non-media-files). However, some apps have a core use case that requires broad access of files on a device, but cannot do so efficiently using the privacy-friendly storage best practices. Android provides a special app access called *All files access* for these situations.

For example, an anti-virus app's primary use case might require regular scanning of many files across different directories. If this scanning requires repeated user interactions to select directories using the system file picker, it may provide a poor user experience. Other use cases—such as file manager apps, backup and restore apps, and document management apps—may require similar considerations.

Request All files access

An app can request All files access from the user by doing the following:

1. Declare the `MANAGE_EXTERNAL_STORAGE` (/reference/android/Manifest.permission#MANAGE_EXTERNAL_STORAGE) permission in the manifest.
2. Use the `ACTION_MANAGE_ALL_FILES_ACCESS_PERMISSION` (/reference/android/provider/Settings#ACTION_MANAGE_ALL_FILES_ACCESS_PERMISSION) intent action to direct users to a system settings page where they can enable the following option for your app: **Allow access to manage all files**.

To determine whether your app has been granted the `MANAGE_EXTERNAL_STORAGE` permission, call

`Environment.isExternalStorageManager()` (/reference/android/os/Environment#isExternalStorageManager()).

Operations that MANAGE_EXTERNAL_ STORAGE allows

The `MANAGE_EXTERNAL_STORAGE` permission grants the following:

- Read and write access to all files within shared storage (/training/data-storage/shared).

★ **Note:** The **`/sdcard/Android/media`** directory is part of shared storage.

- Access to the contents of the `MediaStore.Files` (/reference/android/provider/MediaStore.Files) table.
- Access to the root directory of both the USB on-the-go (OTG) drive and the SD card.
- Write access to all internal storage directories, except for `/Android/data/`, `/sdcard/Android`, and most subdirectories of `/sdcard/Android`. This write access includes direct file path (/training/data-storage/shared/media#direct-file-paths) access.

  Apps that are granted this permission still cannot access the app-specific directories (/training/data-storage/app-specific) that belong to other apps because these directories appear as subdirectories of `Android/data/` on a storage volume.

When an app has the `MANAGE_EXTERNAL_STORAGE` permission, it can access these additional files and directories using either the `MediaStore` (/reference/android/provider/MediaStore) API or direct file paths (/training/data-storage/shared/media#direct-file-paths). When you use the Storage Access Framework (/training/data-storage/shared/documents-files), however, you can only access a file or directory if you could do so without having the `MANAGE_EXTERNAL_STORAGE` permission.

Invoke another app's storage management activity

On Android 12 (API level 31) and higher, apps that have both the `MANAGE_EXTERNAL_STORAGE` (/reference/android/Manifest.permission#MANAGE_EXTERNAL_STORAGE) permission and the `QUERY_ALL_PACKAGES` (/reference/android/Manifest.permission#QUERY_ALL_PACKAGES) permission—such as file management apps—can use the `getManageSpaceActivityIntent()` (/reference/android/os/storage/StorageManager#getManageSpaceActivityIntent(java.lang.String,%20int)) to send users to another app's custom space management activity (/training/data-storage/app-specific#create-storage-management-activity) .

The `getManageSpaceActivityIntent()` method takes in a package name and a request code, and it returns one of the following:

- A `PendingIntent` (/reference/android/app/PendingIntent), if the app with the specified package name has defined a custom "manage space" activity. The file management app that called the `getManageSpaceActivityIntent()` method can then invoke the returned intent to send users to the custom activity.
- `null`, if the app with the specified package name doesn't define a "manage space" activity.

Enable MANAGE_EXTERNAL_STORAGE for testing

To explore how the `MANAGE_EXTERNAL_STORAGE` permission affects your app, you can enable the permission for testing purposes. To do so, run the following command on the machine that's connected to your test device:

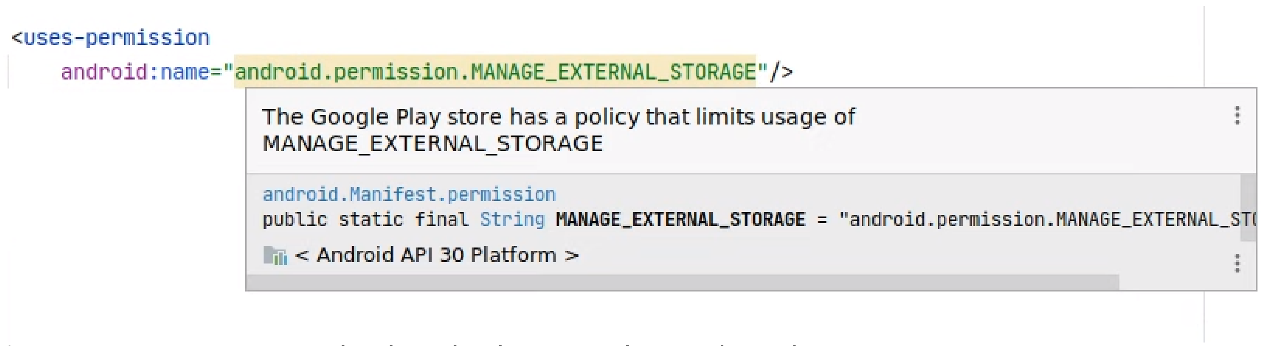
```
$ adb shell appops set --uid PACKAGE_NAME MANAGE_EXTERNAL_STORAGE allow
```

Google Play notice

This section provides a notice for developers who publish apps on Google Play.

To limit broad access to shared storage, the Google Play store has updated its policy (https://support.google.com/googleplay/android-developer/answer/10467955) to evaluate apps that target Android 11 (API level 30) or higher and request "All files access" through the `MANAGE_EXTERNAL_STORAGE` permission. This policy takes effect in May 2021.

When your app targets Android 11 or higher, and it declares the `MANAGE_EXTERNAL_STORAGE` permission, Android Studio shows the lint warning that appears in figure 1. This warning reminds you that "the Google Play store has a policy that limits usage of" the permission.



```
<uses-permission
    android:name="android.permission.MANAGE_EXTERNAL_STORAGE"/>
```

The Google Play store has a policy that limits usage of MANAGE_EXTERNAL_STORAGE

```
android.Manifest.permission
public static final String MANAGE_EXTERNAL_STORAGE = "android.permission.MANAGE_EXTERNAL_ST
```

< Android API 30 Platform >

Figure 1. Lint warning in Android Studio that reminds you about the Google Play policy regarding the `MANAGE_EXTERNAL_STORAGE` permission.

You should request the `MANAGE_EXTERNAL_STORAGE` permission only when your app cannot effectively make use of the more privacy-friendly APIs, such as Storage Access Framework (/training/data-storage/shared/documents-files) or the Media Store API (/training/data-storage/shared/media). Additionally, the app's usage of the permission must fall within permitted uses, and must be directly tied to the core functionality of the app. If your app includes a use case that's similar to the following examples, it's likely to be allowed to request the `MANAGE_EXTERNAL_STORAGE` permission:

- File managers
- Backup and restore apps
- Anti-virus apps
- Document management apps
- On-device file search
- Disk and file encryption
- Device-to-device data migration