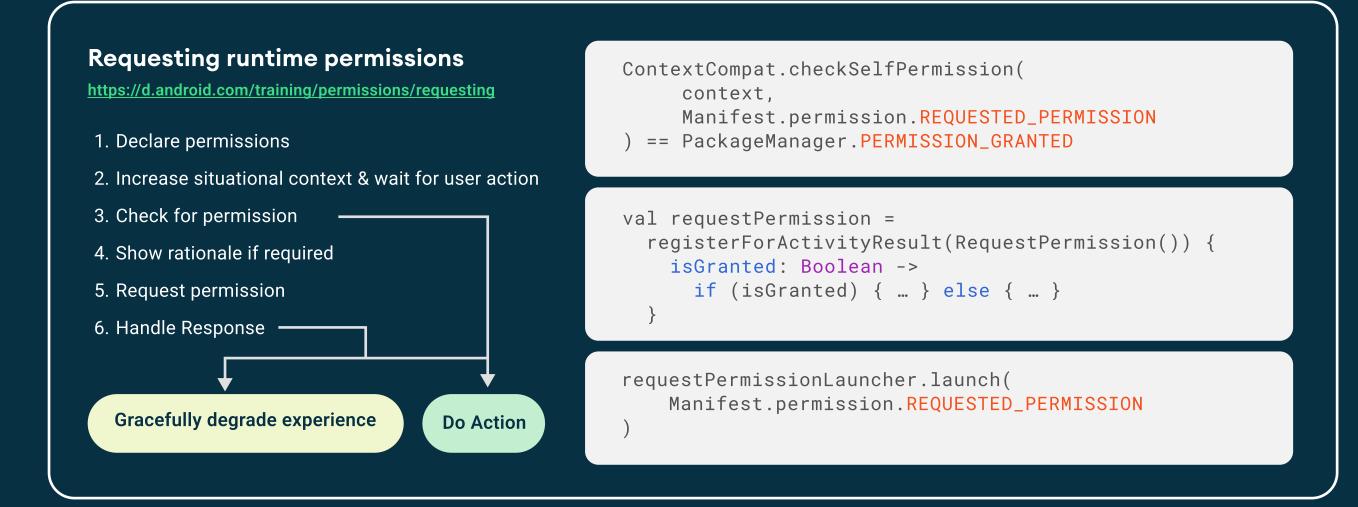
Private by design Android apps

https://d.android.com/design-for-safety





Permission denials

If the user denies a permission, gracefully degrade your app's experience

- Clearly highlight the feature / part of your app with limited functionality due to the permission denial
- Let users continue to use other features in your app normally

Use photo picker

https://d.android.com/training/data-storage/shared/photopicker

Use Photo Picker to provide a seamless and permissonless user experience to select visual media files and simplify developer costs

```
// or PickMultipleVisualMedia(num) for multiple selection
val pickMedia = registerForActivityResult(PickVisualMedia()) { uri ->
   // If media selected URI != null
```

pickMedia.launch(PickVisualMediaRequest(PickVisualMedia.ImageAndVideo))

Choose the right storage

https://d.android.com/training/data-storage

Plan your storage and file locations carefully

- ✓ App-specific storage
- Shared storage
- Preferences
- Databases

Permission minimized requests registerForActivityResult(PickContact()) { uri -> ... } https://d.android.com/training/permissions/evaluating Manage phone Identify device Pair over Bluetooth calls & text

Data Safety Labels

https://d.android.com/guide/topics/data/collect-share

- The Data safety section on Google Play specifies the user data your app collects or shares
- Fill out the form in Play Console (Policy > App content) based on your app and 3rd party SDK data usage.
- Info from your app's <u>Data safety form</u> will be highlighted in your app's Play Store page and for certain critical in-app permission requests.

Understanding where and how your app is accessing user data

https://d.android.com/training/guide/topics/data/audit-access

Use data access audit APIs to gain insights into how your app & its dependencies (including SDKs + 3rd party libraries) access private data from users

Set up callback to log different forms of data access

```
val appOpsManager = getSystemService<AppOpsManager>()
appOpsManager.setOnOpNotedCallback(
   mainExecutor,
    appOpsCallback
```

Set up callback to log different forms of data access

```
val appOpsCallback = object : AppOpsManager.OnOpNotedCallback() {
   override fun onNoted(syncNotedAppOp: SyncNotedAppOp){
     // log synchronous data access (eg. microphone access)
   override fun on AsyncNoted(asyncNotedAppOp: AsyncNotedAppOp){
     // log asynchronous data access (eg. getting location)
   override fun onSelfNoted(syncNotedAppOp: SyncNotedAppOp) {
     // log self data access - fairly rare occurrence}
```

Revoke permissions

Testing permissions

testing purposes.

On Android 13 onward, remove previously granted permission when no longer needed.

More at https://d.android.com/reference/androidx/activity/result/contract/ActivityResultContracts

revokeSelfPermissionOnKill(Manifest.permission.CALL_PHONE

Use ADB commands to grant/revoke permissions for

// Grants all apk permissions

android.permission.CAMERA

android.permission.CAMERA

adb shell pm grant com.name.app/

adb shell pm revoke com.name.app/

adb shell install -g PATH_TO_APK_FILE♪

Package visibility

</manifest>

https://d.android.com/training/package-visibility

Declare package visibility needs when requesting information from other apps (outside of the ones that are visible by default)

```
<manifest package="com.example.game">
  <queries>
  <!-- Specific package names -->
    <package android:name="com.example.store" />
  <!-- Match an intent filter signature -->
    <action android:name="android.intent.action.SEND" />
    <data android:mimeType="image/jpeg" />
   </intent>
  <!-- Specific authority -->
   ovider
    android:authorities="com.example.settings.files" />
   </queries>
```

Location minimization

https://d.android.com/training/location/permissions

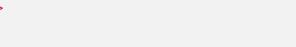
- Always request ACCESS_COARSE_LOCATION only/first
- Ensure your app still works if user grants coarse access only
- Reduce location computation by choosing the best location estimate
- Use **FusedLocationProvider** to ensure compat across form factors (e.g. tablets)
- val locationService = LocationServices.getFusedLocationProviderClient()
- val lastKnownLocation = locationService.getLastLocation()

Background location

https://d.android.com/training/location/background

- Only a few use cases allowed (such as geofencing)
- Before requesting, user must grant **fine** and/or **coarse** permissions first
- Requires **ACCESS_BACKGROUND_LOCATION** permission
- Location frequency might be affected by background location limits

<manifest> <uses-permission</pre> android:name="android.permission.ACCESS_BACKGROUND_LOCATION" /> </manifest>



Scoped storage enhancements

Android 11

- Separate request for background location
- Data access auditing



Android 12

- Approximate location
- Privacy dashboard
- ✓ Bluetooth permissions



Android 13

- Notification permission
- Wi-Fi and storage permissions
- Photo picker



Android 14

- Selected media access
- ✓ Data safety in permissions
- Screenshot detection