Permission

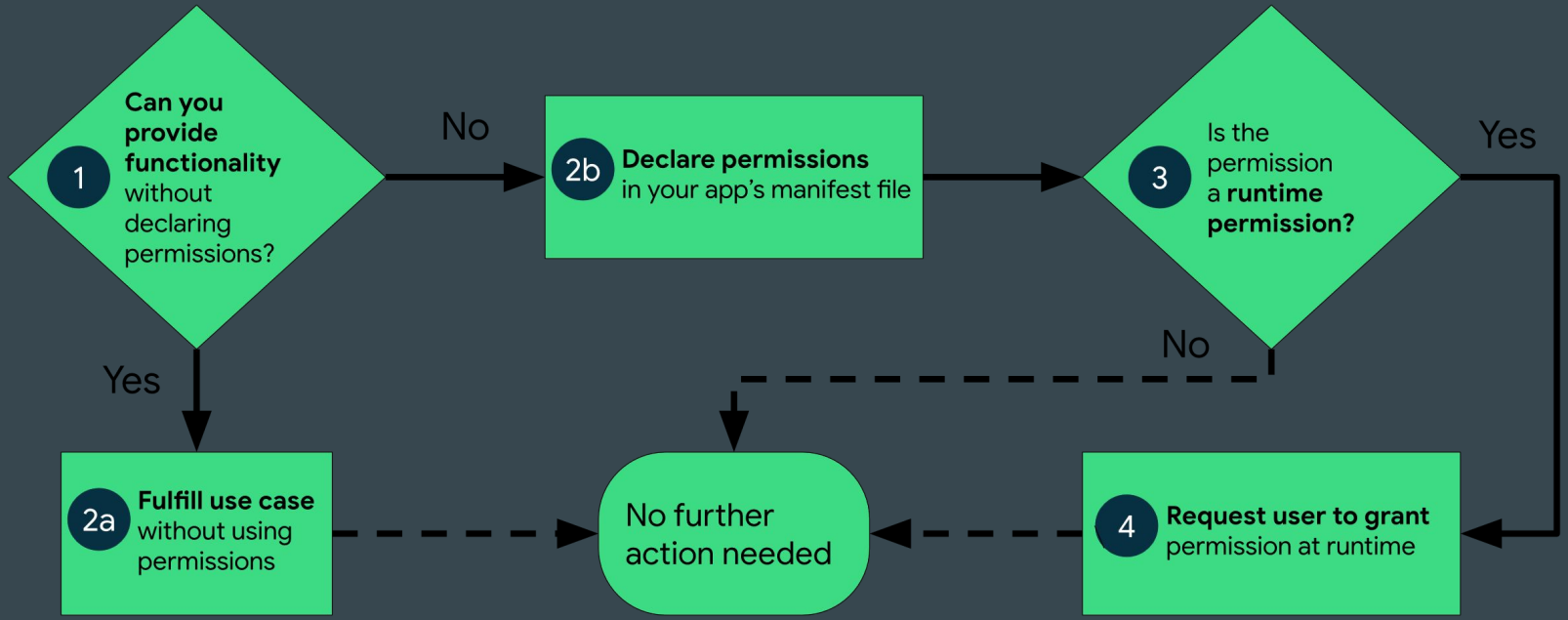
...

Permission

●App permissions help support user privacy by protecting access to the following:

- Restricted data, such as system state and users' contact information
- Restricted actions, such as connecting to a paired device and recording audio

Permission Overview



Permission Types

- Android categorizes permissions into different types, including install-time permissions, runtime permissions, and special permissions.
- Each permission's type indicates the scope of restricted data that your app can access, and the scope of restricted actions that your app can perform, when the system grants your app that permission.

Permission Types

- Install-time permissions give your app limited access to restricted data or let your app perform restricted actions that minimally affect the system or other apps.
- When you declare install-time permissions in your app, an app store presents an install-time permission notice to the user when they view an app's details page.
- The system automatically grants your app the permissions when the user installs your app.
- Android includes several sub-types of install-time permissions, including normal permissions and signature permissions.

Permission Types

- Normal Permission : These permissions allow access to data and actions that extend beyond your app's sandbox but present very little risk to the user's privacy and the operation of other apps.
- Signature Permission : The system grants a signature permission to an app only when the app is signed by the same certificate as the app or the OS that defines the permission.
- Applications that implement privileged services, such as autofill or VPN services, also make use of signature permissions. These apps require service-binding signature permissions so that only the system can bind to the services

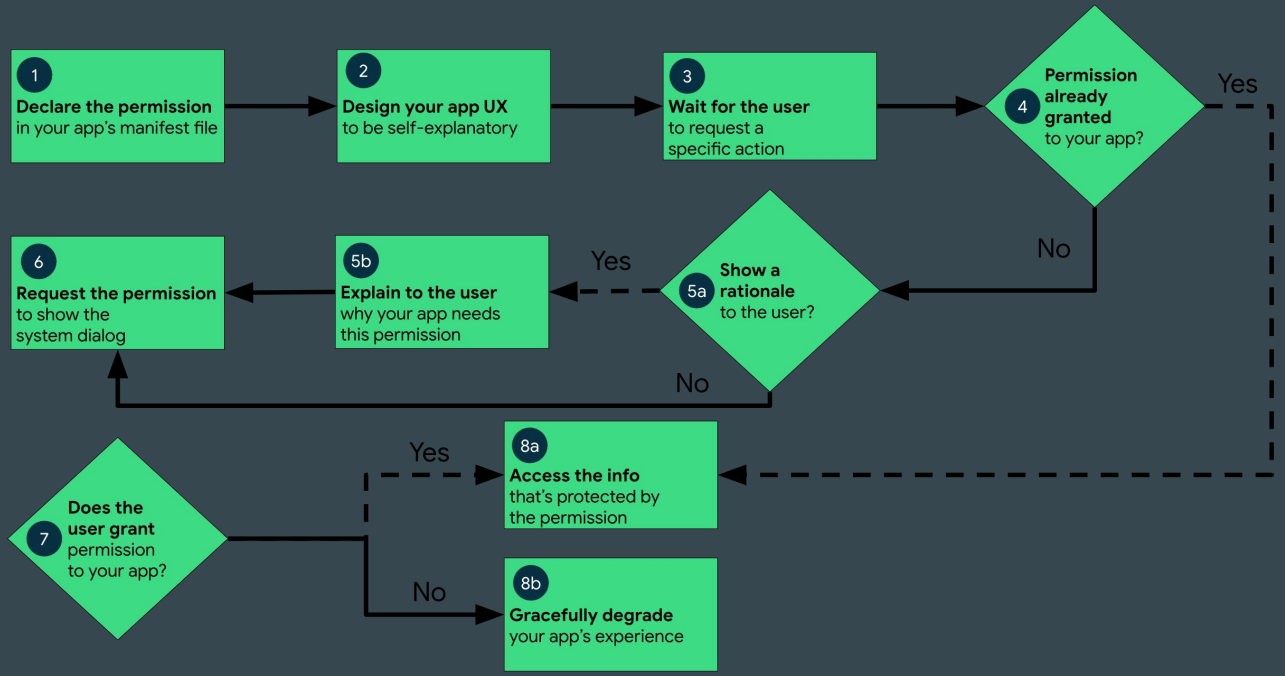
Permission Types

- Runtime permissions, also known as dangerous permissions, give your app additional access to restricted data or let your app perform restricted actions that more substantially affect the system and other apps.
- Therefore, you need to request runtime permissions in your app before you can access the restricted data or perform restricted actions.
- Don't assume that these permissions have been previously granted—check them and, if needed, request them before each access.
- Many runtime permissions access *private user data*, a special type of restricted data that includes potentially sensitive information. Examples of private user data include location and contact information.

Permission

- Special permissions correspond to particular app operations. Only the platform and OEMs can define special permissions. Additionally, the platform and OEMs usually define special permissions when they want to protect access to particularly powerful actions, such as drawing over other apps.
- The **Special app access** page in system settings contains a set of user-toggleable operations. Many of these operations are implemented as special permissions.

Permission Workflow



Permission Workflow Steps

1 In your app's manifest file, declare the permissions that your app might need to request.

2 Design your app's UX so that specific actions in your app are associated with specific runtime permissions. Let users know which actions might require them to grant permission for your app to access private user data.

3 Wait for the user to invoke the task or action in your app that requires access to specific private user data. At that time, your app can request the runtime permission that's required for accessing that data.

4 Check whether the user has already granted the runtime permission that your app requires. If so, your app can access the private user data. If not, continue to the next step.You must check whether you have a permission every time you perform an operation that requires that permission.

Permission Workflow Steps

5A Check whether your app should show a rationale to the user, explaining why your app needs the user to grant a particular runtime permission. If the system determines that your app shouldn't show a rationale, continue to the next step directly, without showing a UI element.

5B If the system determines that your app should show a rationale, however, present the rationale to the user in a UI element. In this rationale, clearly explain what data your app is trying to access and what benefits the app can provide to the user if they grant the runtime permission. After the user acknowledges the rationale, continue to the next step.

Permission Workflow basics

6 Request the runtime permission that your app requires to access the private user data. The system displays a runtime permission prompt, such as the one shown on the permissions overview page.

7 Check the user's response—whether they chose to grant or deny the runtime permission.

8 If the user granted the permission to your app, you can access the private user data. If the user denied the permission instead, gracefully degrade your app experience so that it provides functionality to the user without the information that's protected by that permission.

THE CODE FROM RUNTIMEPERMISSIONBASIC AND IMAGEVIDEO DEMONSTRATION.

Permission

Defining Permissions :

```
<manifest ...>
    <uses-permission android:name="android.permission.CAMERA"/>
    <application ...>
        ...
    </application>
</manifest>
```

Permission

To check whether, permission is already granted or not

ActivityCompat.checkSelfPermission(this, Manifest.permission.permission_name)

To display permission dialog :

ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.permission_name}, permission_code);

this : refers context object, permission_name : for which you need access, permission_code : integer code for permission

Permission

To check whether, rationale should be shown or not :

```
ActivityCompat.shouldShowRequestPermissionRationale(this,
        Manifest.permission.permission_name)
```

Overrided method, to check permission result and perform action :

```
public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions,
        @NonNull int[] grantResults) {}
```

Disclaimer

- All the content is curated from Android Documentation and Google Developer Fundamentals.

- Refer Code Demonstration for details