# Android权限控制内容

Last edited by **caoquanli** 1 month ago

# Android 权限控制内容



## 权限概述

权限的功能目的是保护Android用户的隐私。Android应用必须请求访问敏感用户数据（如联系人和短信）的权限，以及某些系统功能（如相机和互联网）。根据功能的不同，系统可能会自动授予权限，也可能会提示用户批准请求。

## 权限使用

应用必须通过AndroidManifest.xml标记来宣传其所需的权限，在AndroidManifest中使用的权限，在安装的过程中都会保存到data/system/packages.xml.
eg:

```
<uses-permission android: name = “android.permission.SEND_SMS” />
```

## 权限类型

### 1.普通权限：

直接在AndroidManifest.xml中添加就行。
从Android 9（API级别28）开始，以下权限分类为 PROTECTION_NORMAL：

- ACCESS_LOCATION_EXTRA_COMMANDS
- ACCESS_NETWORK_STATE
- ACCESS_NOTIFICATION_POLICY
- ACCESS_WIFI_STATE
- BLUETOOTH
- BLUETOOTH_ADMIN
- BROADCAST_STICKY
- CHANGE_NETWORK_STATE
- CHANGE_WIFI_MULTICAST_STATE
- CHANGE_WIFI_STATE
- DISABLE_KEYGUARD
- EXPAND_STATUS_BAR
- FOREGROUND_SERVICE
- GET_PACKAGE_SIZE
- INSTALL_SHORTCUT
- INTERNET
- KILL_BACKGROUND_PROCESSES
- MANAGE_OWN_CALLS
- MODIFY_AUDIO_SETTINGS
- NFC
- READ_SYNC_SETTINGS

- READ_SYNC_STATS
- RECEIVE_BOOT_COMPLETED
- REORDER_TASKS
- REQUEST_COMPANION_RUN_IN_BACKGROUND
- REQUEST_COMPANION_USE_DATA_IN_BACKGROUND
- REQUEST_DELETE_PACKAGES
- REQUEST_IGNORE_BATTERY_OPTIMIZATIONS
- SET_ALARM
- SET_WALLPAPER
- SET_WALLPAPER_HINTS
- TRANSMIT_IR
- USE_FINGERPRINT
- VIBRATE
- WAKE_LOCK
- WRITE_SYNC_SETTINGS

## 2.签名权限：

系统在安装时授予这些应用程序权限，但只有当试图使用权限的应用程序与定义权限的应用程序签署了相同的证书时才授予这些权限。 从Android 8.1（API级别27）开始，第三方应用程序可以使用的以下权限分类为PROTECTION_SIGNATURE：

- BIND_ACCESSIBILITY_SERVICE
- BIND_AUTOFILL_SERVICE
- BIND_CARRIER_SERVICES
- BIND_CHOOSER_TARGET_SERVICE
- BIND_CONDITION_PROVIDER_SERVICE
- BIND_DEVICE_ADMIN
- BIND_DREAM_SERVICE
- BIND_INCALL_SERVICE
- BIND_INPUT_METHOD
- BIND_MIDI_DEVICE_SERVICE
- BIND_NFC_SERVICE
- BIND_NOTIFICATION_LISTENER_SERVICE
- BIND_PRINT_SERVICE
- BIND_SCREENING_SERVICE
- BIND_TELECOM_CONNECTION_SERVICE
- BIND_TEXT_SERVICE
- BIND_TV_INPUT
- BIND_VISUAL_VOICEMAIL_SERVICE
- BIND_VOICE_INTERACTION
- BIND_VPN_SERVICE
- BIND_VR_LISTENER_SERVICE
- BIND_WALLPAPER
- CLEAR_APP_CACHE
- MANAGE_DOCUMENTS
- READ_VOICEMAIL
- REQUEST_INSTALL_PACKAGES
- SYSTEM_ALERT_WINDOW
- WRITE_SETTINGS
- WRITE_VOICEMAIL

## 3.危险权限：

android6.0以及android6.0以上添加的权限要求，先在AndroidManifest.xml中添加，然后在代码中再次请求权限，我们安装应用后，在Settings中的应用权限里面可以查看到我们的应用有什么危险权限，如果你的应用是在运行在system进程中，或者在system/priv-app目录下，或者targetSdkVersion为22或更低则只需要在AndroidManifest.xml中添加就行。
demo：
第一步在AndroidManifest.xml中添加

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.demo.sunchao.permissiontest">
     <!-- 添加联系人的权限-->
    <uses-permission android:name="android.permission.READ_CONTACTS"/>

    <application
    ...
    </application>
```

```
</manifes>
```

第二步在代码中请求：

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        if (ContextCompat.checkSelfPermission(MainActivity.this, Manifest.
                        permission.CALL_PHONE) != PackageManager.PERMISSION_GRANTED) {
              requestPermissions( new String[] {
                    Manifest.permission.CALL_PHONE},1);
        } else {
            call();
    }

    }

    public void  call() {

    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
        switch (requestCode){
            case 1 :
                if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION
                  call();
                } else {
                    Toast.makeText(this,"You denied the permission",Toast.LENGTH_SHORT).sho
                }
                break;
            default:
        }
    }
}
```

运行后会出现弹窗

## 4.自定义权限:

我们可以自己在AndroidManifest.xml中自定义自己的权限。 eg:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.zlp.myapplication">

    <permission
        android:name="paul.permission.TEST"
        android:description="@string/permdesc_deadlyActivity"
        android:permissionGroup="paul.permission-group.TEST"
        android:protectionLevel="signature" />


    ...
</manifest>
```

## 5.特殊权限:

有一些权限的行为与正常和危险的权限不同。SYSTEM_ALERT_WINDOW和WRITE_SETTINGS特别敏感，所以大多数应用程序不应该使用它们。如果应用程序需要这些权限之一，它必须在清单中声明权限，并发送一个Intent请求用户的授权，系统通过向用户显示详细的管理屏幕来响应Intent。

```
Intent intent = new Intent(android.provider.Settings.ACTION_MANAGE_WRITE_SETTINGS);
intent.setData(Uri.parse("package:" + context.getPackageName()));
intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
context.startActivity(intent);
```

### 6.SignatureOrSystem（系统级权限)：

源码编译完成后的（out/target/common/obj/APPS/framework-res_intermediates/src/android/Manifest.java）中查找，标注有@android.annotation.SystemApi的字符串，就是系统级权限

## 权限等级：

根据protectionLevel来划分为Normal、Dangerous、Signature、SignatureOrSystem 四种等级

### Normal（普通权限）

权限被声明为Normal级别，任何应用都可以申请，在安装应用时，不会直接提示给用户。

### Dangerous（危险权限）

权限被声明为Dangerous级别，任何应用都可以申请，在安装应用时，会直接提示给用户。

### Signature（签名权限）

权限限被声明为Signature级别，理论上定义signature级别权限的应用和使用signature级别权限的应用拥有相同的签名，才可以使用签名权限。 那么只有Android官方使用相同私钥签名的应用才可以申请该权限。但是某些系统定义的签名权限，第三方应用即使签名不相同也可以直接使用，可以看 签名权限

如果App A自定义了一个等级是signature的权限，即android:protectionLevel="signature"，则可以使用该权限的app包括：

（1）与App A有相同签名的app。
（2）在system进程中运行的app。

### SignatureOrSystem（系统级权限）

权限被声明为SignatureOrSystem级别

如果App A自定义了一个等级是signatureOrSystem的权限，即android:protectionLevel="signature|privileged"或者android:protectionLevel="signatureOrSystem"，则可以使用该权限的app包括：

（1）与App A有相同签名的app
（2）在system进程中运行的APP
（3）在/system/priv-app目录中的app，不管是否满足（1）和（2）的条件。即任意app只要放到了/system/priv-app就可以使用App A的signatureOrSystem级别的权限。

## 四大组件权限：

### Actvity 权限：

Activity 在启动的过程中只有权限匹配上了才能启动。有时候我们也可以在添加Activity的AndroidMainfest中去自定义权限的，使用android:permission属性对清单中的标记应用的权限限制了谁可以启动它，如果调用这个Activity没有相关权限使用，则会抛出SecurityException。
eg：
首先在第一个应用的AndroidManifest中定义一个权限:

```
<permission android:name="packagename.permission.Activity"/>
```

1 然后在相应的Activity声明权限并定义一个action（在AndroidManifest中）：

```
<activity
            android:name="Activity"
            android:label="@string/title_activity"
            android:permission="packagename.permission.Activity"
            >
            <intent-filter>
                <category android:name="android.intent.category.DEFAULT"/>
                <action android:name="packagename.intent.action.Activity"/>
```

```
            </intent-filter>
        </activity>
```

在另一个应用中启动该Activity:
首先声明需要的权限:

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

1 然后启动该 Activity:

```
startActivity(new Intent("packgename.intent.action.Activity"));
```

在启一个Activity的过程中，我们需要满足以下条件的一种就可以成功startActivity而不会得到permission denial的异常：

1、同一个application下

2、Uid相同

3、permission匹配

4、目标Activity的属性Android:exported=”true”

5、目标Activity具有相应的IntentFilter，存在Action动作或其他过滤器并且没有设置exported=false

6、启动者的Pid是一个System Server的Pid

7、启动者的Uid是一个System Uid

如果上述调节，满足一条，一般即可（与其他几条不发生强制设置冲突），否则，将会得到Permission Denial的Exception而导致Force Close。

## Service 权限

Service 在启动的过程中，只有权限匹配上了才可以正常启动，不然会抛出权限异常的问题，我们也可以在添加Service的AndroidMainfest中去自定义权限的，使用android:permission属性对清单中的标记应用的权限限制了谁可以启动它，如果startService（），bindService（）或stopService（）的调用者没有被授予此权限，则该方法将不会工作，并且Intent对象不会传递到服务中，则会抛出SecurityException。（具体代码可以参考上面的activity的）

在启一个服务的过程中，我们需要满足以下条件的一种就可以成功启动服务而不会得到permission denial的异常：

1、同一个application下

2、Uid相同

3、permission匹配

4、目标Service的属性Android:exported=”true”

5、目标Service具有相应的action，存在Action动作或其他过滤器并且没有设置exported=false

6、启动者的Pid是一个System Server的Pid

7、启动者的Uid是一个System Uid

如果上述调节，满足一条，一般即可（与其他几条不发生强制设置冲突），否则，将会得到Permission Denial的Exception而导致Force Close。

## Broadcast 权限

**发送权限：** 当您调用sendBroadcast(Intent, String)或 sendOrderedBroadcast(Intent, String, BroadcastReceiver, Handler, int, String, Bundle)，您可以指定权限参数，只有已经请求权限的应用才可以接收广播
eg：

```
sendBroadcast(new Intent("com.example.NOTIFY"),
              Manifest.permission.SEND_SMS);
```

要接收广播，接收应用必须请求权限，如下所示：

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

**接收权限：**
如果您在注册广播接收器时指定了权限参数，那么广播的发送者必须在AndroidManifest中使用请求权限的声明 eg：
假设接收应用程序具有清单声明的接收器，如下所示：

```
<receiver android:name=".MyBroadcastReceiver"
          android:permission="android.permission.SEND_SMS">
    <intent-filter>
        <action android:name="android.intent.action.AIRPLANE_MODE"/>
    </intent-filter>
</receiver>
```

为了能够向这些接收者发送广播，发送应用必须请求权限

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

广播的在发送过程中，有多种权限限制：
1.发送受保护的广播（frameworks/base/core/res/AndroidMainfest.xml）我们必须具有ROOT_UID，SYSTEM_UID，PHONE_UID，BLUETOOTH_UID，NFC_UID，persistent 权限的APK。

2.发送sticky类型的广播，我们需要在AndroidManifest中声明android.Manifest.permission.BROADCAST_STICKY的权限

3.发送ACTION_UID_REMOVED，ACTION_PACKAGE_REMOVED，ACTION_PACKAGE_CHANGED，ACTION_EXTERNAL_APPLICATIONS_UNAVAILABLE， ACTION_EXTERNAL_APPLICATIONS_AVAILABLE，ACTION_PACKAGES_SUSPENDED，ACTION_PACKAGES_UNSUSPENDED类型的广播。我们必须AndroidMainfest中声明android.Manifest.permission.BROADCAST_PACKAGE_REMOVED的权限。

4.拥有system权限的APK去发送一个没有受保护的广播的广播，从log中也会出现异常，但是此广播是可以发送出去的，因此建议系统级APK最好发送受保护的广播。

### ContentProvider权限

在访问数据库的时候，我们需要拥有readPermission，writePermission的权限才可以访问数据库，我们可以自定义ContentProvider由谁来访问的权限，以设置两个单独的权限属性： 限制谁可以从应用程序读取，并限制谁可以应用程序写入它，如果没有声明权限，将会出现SecurityException。
eg：

```
<provider
    android:name=".PeopleContentProvider"
    android:authorities="com.harvic.provider.PeopleContentProvider"
    android:exported="true"
    android:readPermission="com.harvic.contentProviderBlog.read"/>
```

那么我需要在调用者这里申请权限才能使用：

```
<uses-permission android:name="com.harvic.contentProviderBlog.read"/>
```

## 应用安装目录权限区别：

在介绍目录的权限之前先介绍一下什么是system APP？什么是privileged APP（特权APP）？ 什么是普通APP？

### 什么是system APP？

**第一类System APP：**
shared uid为android.uid.system，android.uid.phone，android.uid.log，android.uid.nfc，android.uid.bluetooth，android.uid.shell，android.uid.systemui被视为System APP。

**第二类System APP：**
特定目录包括：/vendor，/system/framework，/system/priv-app，/system/app，/vendor/app，/oem/app。这些目录中的app，被视为system APP。

### 什么是privileged APP（特权APP）？

特权APP首先必须是System APP，也就是说 System APP分为普通的system APP和特权的system APP。特权APP使用危险权限不会弹框，普通的System APP使用危险权限则弹框。

**第一类privileged APP：**
share uid
为"android.uid.system"，"android.uid.phone"，"android.uid.log"，"android.uid.nfc"，"android.uid.bluetooth"，"android.uid.shell"的app。并且拥有platform签名

**第二类privileged APP：** /system/framework和/system/priv-app目录下的app都是特权APP。

### 什么是普通APP?

存放在data/app的目录下面的app都是普通APP。

## 应用安装目录

应用安装目录分为：system/app ， system/priv-app ，vender，data/app 目录，且system/priv-app目录权限等级最高.拥有所有权限（普通权限，危险权限，签名权限，自定义权限，特殊权限，系统级权限）。

### system/app：

无法卸载，无法删除，默认情况下是没有系统级权限,并且使用危险权限是需要弹出提示框。除非APK是特权APP，则可以使用系统级权限，并且使用危险权限不会出现弹窗。

### system/priv-app：

属于特权APP，无法卸载，无法删除。例如：settings,SystemUI, Launcher,都会在 /etc/permissions/privapp-permissions-platform.xml 目录下的系统配置XML文件的白名单中，拥有系统级权限，使用危险权限不会出现弹窗,未在这些XML文件中明确列出的应用不会被授予系统级权限。

### vender：

和system/app一样。

### data/app：

可以卸载，消除数据，无法拥有系统及权限。使用危险权限必须弹框。

## 权限组的定义：

权限被组织成与设备功能或特性相关的组。在该系统中，权限请求在组级别处理，单个权限组对应于APP manifest中的多个权限声明。例如，SMS组包括READ_SMS和RECEIVE_SMS声明。以这种方式对权限进行分组可以使用户做出更

有意义和更明智的选择，而不会被复杂的技术权限请求所压倒。

当您的应用请求危险权限时，以下系统行为适用：

- 如果应用程序当前在权限组中没有任何权限，系统会向用户显示描述应用程序要访问的权限组的权限请求对话框。该对话框未描述该组中的特定权限。例如，如果某个应用请求了该 READ_CONTACTS权限，系统对话框就会说该应用需要访问该设备的联系人。如果用户授予批准，系统将为应用程序提供其请求的权限。
- 如果应用程序已在同一权限组中被授予其他危险权限，则系统会立即授予权限，而不与用户进行任何交互。例如，如果某个应用程序先前已请求并已获得该READ_CONTACTS权限，然后它会请求WRITE_CONTACTS，则系统会立即授予该权限，而不向用户显示权限对话框.
  危险权限和权限组：

| 权限组 | 危险权限 |
| --- | --- |
| CALENDAR | READ_CALENDAR ，WRITE_CALENDAR |
| CALL_LOG | READ_CALL_LOG， WRITE_CALL_LOG， PROCESS_OUTGOING_CALLS |
| CAMERA | CAMERA |
| CONTACTS | READ_CONTACTS，WRITE_CONTACTS，GET_ACCOUNTS |
| LOCATION | ACCESS_FINE_LOCATION，ACCESS_COARSE_LOCATION |
| MICROPHONE | RECORD_AUDIO |
| PHONE | READ_PHONE_STATE，READ_PHONE_NUMBERS， CALL_PHONE，<br>ANSWER_PHONE_CALLS， ADD_VOICEMAIL， USE_SIP |
| SENSORS | BODY_SENSORS |
| SMS | SEND_SMS，RECEIVE_SMS，READ_SMS，RECEIVE_WAP_PUSH，RECEIVE_MMS |

| 权限组 | 危险权限 |
| --- | --- |
| STORAGE | READ_EXTERNAL_STORAGE，WRITE_EXTERNAL_STORAGE |

## 查看权限的相关命令

查看设备所有的权限
adb shell pm list permissions -s

查看你某个包的权限
adb shell dumpsys package xxxx

通过dumpsys package 获取与permission相关的信息：

dumpsys package perm|more 或者 dumpsys package permissions | more

查看设备上存在的权限组

pm list permission-groups

查看设备上已经有的权限，包括系统定义和应用自定义

adb shell pm list permissions

按组查看详细的权限

adb shell pm list permissions -g|more

显示权限详细信息

adb shell pm list permissions -f|more

显示权限摘要

adb shell pm list permissions -s|more

只显示dangerous权限

adb shell pm list permissions -d

dangerous和Normal权限

adb shell pm list permissions -u

显示某一个权限的详细信息

dumpsys package permission android.permission.CALL_PHONE|more

查看某个package是否有特定的permission

dumpsys package check-permission com.gionee.cloud.permission.SEND com.google.android.webview