Appendix A

Text Companion

This is the online text companion to the book *Pro Android with Kotlin*.

Manifest Top-Level Entries

This section of the online text companion describes the `<manifest>` entry as used inside file `AndroidManifest.xml`.

The <application> Element

Specifies the application. Because this is the main entry for the application declaration, this is described in detail in the section "The Application Declaration" later in the chapter.

The <compatible-screens> Element

Specifies compatible screens. This is informational only for the app, but it dramatically influences the user base that can install your app using the Google Play store. You should not normally use this attribute but instead provide alternative layouts and bitmaps, as described in Chapter 9. Here's the syntax:

```
<compatible-screens>
  <screen android:screenSize=
        ["small" | "normal" | "large" | "xlarge"] android:screenDensity=
     ["ldpi" | "mdpi" | "hdpi" | "xhdpi" | "280" |
      "360" | "420" | "480" | "560" ] />
    <screen ...
</compatible-screens>
```

Both `screenSize` and `screenDensity` are required for each `<screen>` element.

P. Späth, *Pro Android with Kotlin*, https://doi.org/10.1007/978-1-4842-3820-2

The <instrumentation> Element

Can be used for monitoring interactions. It contains the fully qualified name of a subclass of `android.app.Instrumentation`. We more thoroughly describe instrumentation in Chapter 15. Here's the syntax:

```
<instrumentation android:functionalTest=["true" | "false"]
             android:handleProfiling=["true" | "false"]
             android:icon="drawable resource"
             android:label="string resource"
             android:name="string"
             android:targetPackage="string"
             android:targetProcesses="string" />
```

These are the attributes are in detail:

- **android:functionalTest**

  One of: `true`, `false`. Specifies whether the instrumentation class is to be run as a functional test. The default is `false`.

- **android:handleProfiling**

  One of: `true`, `false`. Specifies whether the instrumentation class will handle turning profiling on and off by itself. If `false`, profiling will run all the time. The default is `false`.

- **android:icon**

  Specifies the resource ID of a4n icon to use for the instrumentation class.

- **android:label**

  Specifies the resource ID of a label to use for the instrumentation class.

- **android:name**

  Specifies the fully qualified name of the class extending the base class `android.app.Instrumentation`. If it starts with a dot (`.`), the package name from the `<manifest>` element gets prepended.

- **android:targetPackage**

  Provides the package name of the app you want to be monitored.

- **android:targetProcesses**

  Added in API level 26. You can use this to specify the processes to run the instrumentation against. Write a comma-separated list of process names here and use an asterisk (*) as a wildcard for all processes of the app defined by `android:targetPackage`. The default is to monitor only the main process of the app defined by `android:targetPackage`.

The <permission> Element

Specifies a custom, nonsystem permission to limit access to components, features, or data that to undiscerningly use implies a security risk. See Chapter 7 for more information. Here's the syntax:

```
<permission android:description="string resource"
        android:icon="drawable resource"
        android:label="string resource"
        android:name="string"
        android:permissionGroup="string"
        android:protectionLevel=["normal" | "dangerous" |
                "signature" | "signatureOrSystem"] />
```

The attributes of this element are as follows:

- **android:description**

  The string resource ID of a user-readable description.

- **android:icon**

  The drawable resource ID of an icon to use for the permission.

- **android:label**

  The resource ID of a label for the permission.

- **android:name**

  The name of the permission. To avoid clashes, it is strongly recommended to prepend the package name.

- **android:permissionGroup**

  The permission group this permission is supposed to be part of. This must match with the name of a `<permission-group>` element.

- **android:protectionLevel**

  A protection level; one of these values:

  - `normal`: This is the default. Signifies a lower risk with permission automatically granted by the system upon installation. The user can investigate such permissions, though.
  - `dangerous`: A higher risk, and the system will usually require a confirmation before accordingly protected features can be used.
  - `signature`: Permission gets granted only if the app requesting the permission is signed with the same certificate as the app declaring the permission.
  - `signatureOrSystem`: Same as `signature` but also includes Android system image apps as requestors. You should not use it.

Note that this element *specifies* a permission; it does not impose it. Imposing permissions happens on a component level.

The <permission-group> Element

Used for grouping permissions in the UI presented to the user. Contained permissions will have the name of the group added as a `permissionGoup` attribute. You can find more details in Chapter 7. Here's the syntax:

```
<permission-group android:description="string resource"
                  android:icon="drawable resource"
                  android:label="string resource"
                  android:name="string" />
```

The attributes are as follows:

- **android:description**

  A string resource ID for a description of this group

- **android:icon**

  A drawable resource ID for an icon

- **android:label**

  A string resource ID for a label

- **android:name**

  The name of the group, to be added to `<permission>` elements

The <permission-tree> Element

Contains the base name for a permission tree used to gather individual permissions. If, for example, you have permissions like `com.abc.permission.allow1` and `com.abc.permission.allow2`, a permission tree with the name `com.abc.permission` will automatically gather these two. You can find more details in Chapter 7. Here's the syntax:

```
<permission-tree android:icon="drawable resource"
                 android:label="string resource" ]
                 android:name="string" />
```

For the attributes, you can use the following:

- **android:icon**

  A drawable resource ID for an icon.

- **android:label**

  A string resource ID for a label.

- **android:name**

  The name of the tree, to be matched against the start of names of the `<permission>` elements. This must contain at least three parts separated by dots (`.`).

The <supports-gl-texture> Element

Zero, one, or more elements of this type tell external parties, most prominently the Google Play store, which OpenGL texture formats the app supports. This is informational to the app; it will not decide anything based on that information. Here is the syntax:

```
<supports-gl-texture
      android:name="string" />
```

Possible values are as follows:

- **GL_OES_compressed_ETC1_RGB8_texture**

  For Ericsson texture compression (OpenGL ES 2.0)

- **GL_OES_compressed_paletted_texture**

  For generic paletted texture compression

- **GL_AMD_compressed_3DC_texture**

  For ATI 3Dc texture compression

- **GL_AMD_compressed_ATC_texture**

  For ATI texture compression (devices running Adreno GPU)

- **GL_EXT_texture_compression_latc**

  For luminance alpha texture compression

- **GL_EXT_texture_compression_dxt1**

  For S3 DXT1 texture compression (Nvidia Tegra2)

- **GL_EXT_texture_compression_s3tc**

  For S3 texture compression (Nvidia Tegra2)

- **GL_IMG_texture_compression_pvrtc**

  For PowerVR texture compression (PowerVR SGX530/540 GPU)

The <supports-screens> Element

Specifies the possible screen sizes your app supports. Here's the syntax:

```
<supports-screens android:resizeable=["true"| "false"]
            android:smallScreens=["true" | "false"]
            android:normalScreens=["true" | "false"]
            android:largeScreens=["true" | "false"]
            android:xlargeScreens=["true" | "false"]
            android:anyDensity=["true" | "false"]
            android:requiresSmallestWidthDp="integer"
            android:compatibleWidthLimitDp="integer"
            android:largestWidthLimitDp="integer"/>
```

Here are the attributes explained:

- **android:resizeable**

  One of: `true`, `false`. Specifies whether the app is resizable for different screens. The default is `true`, and because it is deprecated, you should not normally use this attribute.

- **android:smallScreens**

  One of: `true`, `false`. Specifies whether to support small screens (below HVGA). The default is `true`.

- **android:normalScreens**

  One of: `true`, `false`. Specifies whether to support normal screens (HVGA medium, WQVGA low, WVGA high). The default is `true`.

- **android:largeScreens**

  One of: `true`, `false`. Specifies whether to support large screens (significantly higher than handset screens). The default is undefined, so you should explicitly set this attribute.

- **android:xlargeScreens**

  One of: `true`, `false`. Specifies whether to support extra-large screens (significantly higher than large screens). The default is undefined, so you should explicitly set this attribute.

- **android:anyDensity**

  One of: `true`, `false`. Tells whether the app can handle different densities by providing different resource sets. The default is `true`, and you should not change it.

- **android:requiresSmallestWidthDp**

  If the smallest screen rectangle your app can handle is PxQ in units of dp, you can set whichever of P and Q is smaller as a value of this attribute. This will not target your app but will target Google Play store filtering (although the availability of this filter is uncertain). You normally don't need this attribute.

- **android:compatibleWidthLimitDp**

  If the smaller side of a device's screen is larger than the value here, you let the user decide whether they want to use the *screen compatibility mode*. This is a radio button in the action bar for selecting either to stretch the layout (makes empty areas bigger) or to perform a zooming (with possible pixel artifacts). You normally don't need this attribute if you designed the app with different screen sizes covered by different resource sets.

- **android:largestWidthLimitDp**

  This is the same as `android:compatibleWidthLimitDp`, but the user doesn't have a choice; a zooming in will happen.

The <uses-configuration> Element

Use this to tell your app it needs a certain way for a user to input data or perform navigation. You should not normally use this attribute and design your app agnostic of the way the user physically interacts with it. Here's the syntax:

```
<uses-configuration
    android:reqFiveWayNav=["true" | "false"]
    android:reqHardKeyboard=["true" | "false"]
    android:reqKeyboardType=["undefined" | "nokeys" |
                            "qwerty" | "twelvekey"]
    android:reqNavigation=["undefined" | "nonav" |
                            "dpad" | "trackball" | "wheel"]
    android:reqTouchScreen=["undefined" | "notouch" |
                            "stylus" | "finger"] />
```

The attributes are as follows:

- **android:reqFiveWayNav**

  One of: `true`, `false`. Specifies whether five-way navigation must exist.

  This is a control for navigating up, down, right, and left, as well as a button for invoking the currently selected item. The default is `false`.

- **android:reqHardKeyboard**

  One of: `true`, `false`. Specifies whether a hardware keyboard is required. The default is `false`.

- **android:reqKeyboardType**

  The keyboard type. This can be one of the following:

  - `undefined`: The default; no special requirement
  - `nokeys`: Keyboard not required
  - `qwerty`: Standard QWERTY keyboard required
  - `twelvekey`: A phone-like keyboard (0 to 9, *, #)

- **android:reqNavigation**

  A navigation requirement. One of:

  - `undefined`: The default; no special requirement
  - `nonav`: No navigation required
  - `dpad`: D-pad required
  - `trackball`: Trackball required
  - `wheel`: Wheel required

- **android:reqTouchScreen**

  A requirement for a touchscreen. One of:

  - `undefined`: The default; no special requirement
  - `notouch`: No touchscreen required
  - `stylus`: Stylus touchscreen required
  - `finger`: Finger touchscreen required

The <uses-feature> Element

Use this to declare a hardware or software feature your app requires. This has no impact on the app but serves as a filter criterion in the Google Play store. Here's the syntax:

```
<uses-feature
      android:name="string"
      android:required=["true" | "false"]
      android:glEsVersion="integer" />
```

For the attributes, you can use the following:

- **android:name**

  The standardized name of the feature. For a list see the "Features" section of this online text companion.

- **android:required**

  One of: `true`, `false`. Specifies whether your app will not work without that feature. Using `false` here means the feature is nice to have, but your app will work without it. The default is `true`.

- **android:glEsVersion**

  Specifies an OpenGL ES version. The higher 16 bits specify the major version code; the lower 16 bits specify the minor version code. So, version 2.0 maps to 0x00020000. The default is to require only version 1.0.

The <uses-permission> Element

Use this to state that your app needs certain permissions to do its work. Up to Android 5.1, permissions get inquired from the user at installation time; starting with Android 6.0, this happens at runtime. Here's the syntax:

```
<uses-permission android:name="string"
        android:maxSdkVersion="integer" />
```

The attributes of this element are as follows:

- **android:name**

  The standardized name of the permission. This contains system and custom permissions (see the previous `<permission>` element). For a list of system permissions, see the "Permissions" section of this online text companion.

- **android:maxSdkVersion**

  Introduced in API level 19. Use this to specify in the latest API level that a permission explicitly is needed by the app.

The <uses-permission-sdk-23> Element

Same as `<uses-permission>`, but valid only if the device is running on API level 23 (Android 6.0) or higher. Here's the syntax:

```
<uses-permission-sdk-23 android:name="string"
        android:maxSdkVersion="integer" />
```

The <uses-sdk> Element

An element specifying the Android API level conformance of your app. This is rather important, because it tells on which Android versions your app will be running and thus defines the possible user base. Here's the syntax:

```
<uses-sdk android:minSdkVersion="integer"
          android:targetSdkVersion="integer"
          android:maxSdkVersion="integer" />
```

The attributes are as follows:

- **android:minSdkVersion**

  The minimum API level a device must have to run your app. If you make this too low, inside the code you have to take care of too many old Android versions. If you make this too high, you lose too many possible users. See Table A-1 to help you to make a correct decision.

- **android:targetSdkVersion**

  You write an API level number into this attribute to express that you developed and tested your app with exactly this number. The Android OS on a device with an API level higher than that will then do its best to make sure your app will run there as well, applying compatibility modes where necessary. It is very reasonable to use this attribute and always write the latest available API level of your test devices or emulated devices you use for development and testing.

- **android:maxSdkVersion**

  The maximum API level your app will run in. You usually don't need this attribute, and you should not use it unless you are okay losing users.

> **Note** Despite the name, these settings address the *API level*, not actually the SDK version.

Table A-1. *API Level Cumulated Distribution (as of the First Quarter of 2018)*

Minimum API Level	Android Version	Distribution
10	2.3	> 99%
15	4.0	> 99%
16	4.1	> 99%
17	4.2	> 95%
18	4.3	> 90%
19	4.4	> 90%
21	5.0	> 70%
22	5.1	> 60%
23	6.0	> 35%
24	7.0	> 8%
25	7.1	> 1%
26	8.0	< 1%
27	8.1	< 1%

The Application Declaration

An Android app is a container for the components building up the application. It is described by the `<application>` element as a child element of `<manifest>` inside `AndroidManifest.xml`. The synopsis of the `<application>` element reads as follows:

```
<application ...>
    <activity ...>...</activity>
    <activity-alias ...>...</activity-alias>
    <meta-data ...>...</meta-data>
    <service ...>...</service>
    <receiver ...>...</receiver>
    <provider ...>...</provider>
    <uses-library ...>...</uses-library>
</application>
```

Table A-2 lists possible attributes of the `<application>` element; the child elements are described in the following paragraphs.

Table A-2. Manifest Application Attributes

Name	Description
android: allowTaskReparenting	One of: true, false. Specifies whether the activities of this app can have their parent activity changed to the value given by the taskAffinity attribute. For example, starting a web page from an e-mail with re-parenting to the browser means the new web page gets attributed to the browser, so the next time the browser get activated, this page will be shown. This makes sense only if the launchMode attribute of the activities is set to standard or singleTop. The default is false.
android: allowBackup	One of: true, false. If true (the default), allows the app to participate in the backup infrastructure of the Android OS.
android: allowClearUserData	One of: true, false. Not available to non-system-image apps. If true (the default for system image apps), the app may reset user data.
android: backupAgent	The fully qualified class name of a backup agent (if starting with a ., the package name gets prepended) extending android.app.backup.BackupAgent.
android: backupInForeground	One of: true, false. If true, an automatic backing up happens while the app is in foreground mode. Use with caution since the app gets shut down from the Android OS during the backing-up operation. The default is false.
android: banner	For Android TV apps, the resource ID of a banner to show on the home screen. There is no default banner.
android: debuggable	One of: true, false. Specifies whether the app can be debugged. The default is false.
android: description	User-readable longer description of the app, as a resource ID.
android: directBootAware	One of: true, false. Specifies whether the app can run before the user unlocks the device. The default is false.
android: enabled	One of: true, false. If false, the app is disabled. The default is true.
android: extractNativeLibs	One of: true, false. If false, do not extract native libraries to the file system while installing the app. The native libraries must then be page aligned and stored uncompressed in the APK file. The default is true.
android: fullBackupContent	Points to an XML file describing what to back up. The syntax for this file is described in the online documentation in the section "Auto Backup for Apps."
android: fullBackupOnly	For Android starting with API level 23. One of: true, false. If true, the device participates in Auto Backup with data synchronized on the Google cloud. The default is false.

(*continued*)

Table A-2. (*continued*)

Name	Description
android: hasCode	One of: true, false. Specifies whether the app contains any code. The default is true.
android: hardwareAccelerated	One of: true, false. If true, enables graphics hardware acceleration. The memory consumption might be higher. The default is false.
android: icon	The resource ID of an icon to be used for the app, for example @mipmap/myapp_icon.
android: isGame	One of: true, false. Specifies whether the app is a game. The default is false.
android: killAfterRestore	One of: true, false. Specifies whether to kill the app after a full system restore operation. The default is true, and you will not normally change it.
android: largeHeap	One of: true, false. A last-resort setting if your app needs a lot of memory. The default is false.
android: label	The resource ID of a label string to be used for the app, for example @string/myapp_label.
android: logo	A resource ID of the app's logo, and a default for the activities included.
android: manageSpaceActivity	The fully qualified name of an activity class (extending android.app.activity) that lets the user administer memory usage of the app. The Android OS might then call it in case of a resource shortage. The default is to not provide such an activity.
android: name	The fully qualified name of a subclass of android.app.Application. You may use it for extra investigation of app activities and for special app-related administration purposes. The default is to use the standard class.
android: networkSecurityConfig	Added in API level 24. The resource ID of an XML file containing the network security configuration. The syntax of this file is available in the online documentation in the "Network Security Configuration" section.
android: permission	The name of a permission the caller needs to be able to call components of this app. If not given here and also not in child elements, no restriction applies.
android: persistent	One of: true, false. Specifies whether the app is supposed to be running at all times. This is left for system-level apps, and you should not use it. The default is false.

(*continued*)

Table A-2. (*continued*)

Name	Description
`android: process`	Normally a component starts in the process of the app it is defined in. If you use this attribute, the process given by the value supplied here is used instead for all components of this app, unless you define the `process` attribute on a component basis as well. If the name starts with a colon (`:`) a new private process gets created for that aim. If it starts with a lowercase letter, a global process with that name will be used instead.
`android: restoreAnyVersion`	One of: `true`, `false`. Specifies whether the backup mechanism is allowed to restore the app even with the backup data coming from newer versions of the app. The default is `false`.
`android: requiredAccountType`	Added in API level 18. The name of an account authenticator type necessary for this app to start. This corresponds to an online account. For restricted profiles, this effectively disables the app because they cannot add accounts. This defaults to not requiring an account.
`android: resizeableActivity`	Added in API level 24. One of: `true`, `false`. If set to `true`, the activities of this app can participate in a multiwindow mode, provided the device is capable of doing that. Otherwise, the activities use the whole screen. The default value is `true`.
`android: restrictedAccountType`	An addition to the `requiredAccountType` attribute earlier; also add account types here if you want restricted profiles to be able to use the app. Caution: this imposes a security risk if the account is connected with personal user data.
`android: supportsRtl`	Added in API level 17. One of: `true`, `false`. Specifies whether to allow a right-to-left layout. The default is `false`.
`android: taskAffinity`	Normally all activities of an app have the same affinity, which means they all go to the same task stack once active. Their affinity is then determined by the root activity's name. If you set an app's `taskAffinity`, though, you can express the activities of this app to belong to another task stack, which takes effect once the activity is re-parented or launched with the `FLAG_ACTIVITY_NEW_TASK` flag set. If unset and also not set in the child elements, take the package name.
`android: testOnly`	One of: `true`, `false`. Set it to `true` if you want to use this app only for testing purposes. It then cannot be published at the Google Play store. The default is `false`.
`android: theme`	Points to a style resource to be used as the app's style theme. If unset and also not set inside the child elements, take the default system theme.

(*continued*)

Table A-2. (*continued*)

Name	Description
`android: uiOptions`	Extra UI options to use. Either `none` for nothing extra (default) or `splitActionBarWhenNarrow` to have an additional action bar at the bottom of the screen when horizontal space gets scarce.
`android: usesCleartextTraffic`	Added in API level 23. Ignored for API level 24 and above if a *Network Security Config* is present. One of: `true`, `false`. If set to `false`, platform components might refuse to transport cleartext data over the network, on a best-effort basis. The default is `true`.
`android: vmSafeMode`	One of: `true`, `false`. Specifies whether the virtual machine should operate in safe mode. The default is `false`.

Note For simple apps, the `<application>` element may contain no attributes and have just a single `<activity>` element. Of course, you want to add attributes for a more professional-looking app, and you will have to add more child elements if your app starts getting more complex.

The <activity> Child Element

Describes a user interface component. You can have zero, one, or more of this element in an application. Activities are thoroughly described in Chapter 3.

The <activity-alias> Child Element

An alias to an activity from the same application. The alias needs to be declared *after* the target activity. Here's the syntax:

```
<activity-alias android:enabled=["true" | "false"]
                android:exported=["true" | "false"]
                android:icon="drawable resource"
                android:label="string resource"
                android:name="string"
                android:permission="string"
                android:targetActivity="string" >
    ...
</activity-alias>
```

The attributes are as follows:

- **android:enabled**

  One of: `true`, `false`. Specifies whether this alias is enabled. The default is `true`.

- **android:exported**

  One of: `true`, `false`. Specifies whether this alias is exported, i.e., usable from other apps. The default is `false` if the element contains no intent filters, otherwise `true`.

- **android:icon**

  The resource ID of an icon for this alias.

- **android:label**

  The string resource ID of a label for this alias.

- **android:name**

  The name of this activity alias. This should follow the same naming convention as for activities, but there is actually no restriction for alias names.

- **android:permission**

  The name of a permission a client needs to have to start the activity declared in attribute `targetActivity`. This defaults to no restriction.

- **android:targetActivity**

  Points to the activity this alias refers to. Enter the activity's name here.

The following are possible children elements:

- **intent-filter**

  See Chapter 3 for a description.

- **meta-data**

  An arbitrary name-value pair in the form `<meta-data android:name="..." android:resource="..." android:value="..." />`. You can have several of them, and they go into an `android.os.Bundle` element available as `PackageItemInfo.metaData`.

The <meta-data> Child Element

An arbitrary name-value pair in the form `<meta-data android:name="..." android:resource="..." android:value="..." />`. You can have several of them, and they go into an `android.os.Bundle` element available as `PackageItemInfo.metaData`.

The <provider> Child Element

This describes a *content provider*, which is a component that provides structured data to other components and apps. It is described in more detail in Chapter 6; here we just summarize the element. You can declare several of them, and one element has the following syntax:

```
<provider android:authorities="list"
          android:directBootAware=["true" | "false"]
          android:enabled=["true" | "false"]
          android:exported=["true" | "false"]
          android:grantUriPermissions=["true" | "false"]
          android:icon="drawable resource"
          android:initOrder="integer"
          android:label="string resource"
          android:multiprocess=["true" | "false"]
          android:name="string"
          android:permission="string"
          android:process="string"
          android:readPermission="string"
          android:syncable=["true" | "false"]
          android:writePermission="string" >
    ...
</provider>
```

The possible attributes are as follows:

- **android:authorities**

  A semicolon-separated list of URI authorities. Although not strictly prescribed, each authority should have a reverse-domain Java-style name like `com.example.peopleprovider.ThePeople`, maybe reflecting the class that extends the `android.content.ContentProvider` subclass. There is no default; you must provide at least one.

- **android:directBootAware**

  One of: `true`, `false`. Specifies whether contents gets provided before the user unlocks the device. The default is `false`.

- **android:enabled**

  One of: `true`, `false`. Specifies whether this provider is enabled. The default is `true`.

- **android:exported**

  One of: `true`, `false`. Specifies whether this provider is exported, i.e., usable from other apps. The default is `false` if you set `android:targetSdkVersion` to 17 or higher and the device runs on API level 17 or higher. Lower API levels run as if it is set to `true`.

- **android:grantUriPermissions**

  One of: `true`, `false`. For any client not listed in a `grant-uri-permission` subelement, if you set this attribute to `true`, permissions can still be granted on a temporary basis by asking the user. The caller must then set flags like `FLAG_GRANT_READ_URI_PERMISSION` or `FLAG_GRANT_WRITE_URI_PERMISSION` in the calling intent. The default is `false`.

- **android:icon**

  The resource ID of an icon for this provider.

- **android:initOrder**

  If inside a process you need an instantiation order for content providers, you can set an appropriate integer value here. Higher numbers come first.

- **android:label**

  The string resource ID of a label for this alias.

- **android:multiprocess**

  One of: `true`, `false`. If your app runs in several processes, by setting this to `true` you can allow several instances of one content provider to be instantiated, one per process that needs it. The default is `false`.

- **android:name**

  The class name of this provider. The class must extend the `android.content.ContentProvider` base class. If it starts with a dot (`.`), the package name gets prepended.

- **android:permission**

  The name of a permission a client needs to have to use this content provider. This is a shortcut for using both `readPermission` and `writePermission` with a single value (the latter has precedence, though).

- **process**

  Normally a component starts in the process of the app it is defined in. If you use this attribute, the process given by the value supplied here is used instead for the content provider. If the name starts with a colon (`:`), a new private process gets created for that aim. If it starts with a lowercase letter, a global process with that name will be used instead.

- **android:readPermission**

  The name of a permission a client needs to have to read from this content provider.

- **android:syncable**

  One of: `true`, `false`. Specifies whether the data can be synchronized with the data on a server.

- **android:writePermission**

  The name of a permission a client needs to have to write to this content provider.

Applicable subelements of this element are as follows:

- **grant-uri-permission**

  An element specifying a URI permission or permission pattern. It has the following form:

```
<grant-uri-permission android:path="string"
    android:pathPattern="string"
    android:pathPrefix="string" />
```

Here, the `path` attribute specifies a complete path, the `pathPrefix` attribute allows matching the initial part of a path, and `pathPattern` is a complete path but with wildcards (*matches zero to many occurrences of the preceding character, and `.*` matches zero to many occurrences of any character).

- **meta-data**

  An arbitrary name-value pair in the form `<meta-data android:name="..." android:resource="..." android:value="..." />`. You can have several of them, and they go into an `android.os.Bundle` element available as `PackageItemInfo.metaData`.

- **path-permission**

  To define a subset of data that a content provider can serve, you can use this element to specify a path and a required permission. It has the following form:

```
<path-permission android:path="string"
    android:pathPrefix="string"
    android:pathPattern="string"
    android:permission="string"
    android:readPermission="string"
    android:writePermission="string" />
```

Here, the `path` attribute specifies a complete path, the `pathPrefix` attribute allows matching the initial part of a path, and `pathPattern` is a complete path but with wildcards (* matches zero to many occurrences of the preceding character, and `.*` matches zero to many occurrences of any character). The `permission` attribute specifies a read and write permission, and the attributes `readPermission` and `writePermission` draw a distinction between read and write permissions. If any of the latter two is specified, it takes precedence over the `permission` attribute.

The <receiver> Child Element

This element (or several of them) specifies *broadcast receivers*. We talk about broadcasts in Chapter 5; here we just provide a summary. The syntax of the element is as follows:

```
<receiver android:directBootAware=["true" | "false"]
          android:enabled=["true" | "false"]
          android:exported=["true" | "false"]
          android:icon="drawable resource"
          android:label="string resource"
          android:name="string"
          android:permission="string"
          android:process="string" >
    ...
</receiver>
```

It has the following attributes:

- **android:directBootAware**

  One of: `true`, `false`. Specifies whether broadcasts can be received before the user unlocks the device. The default is `false`.

- **android:enabled**

  One of: `true`, `false`. Specifies whether the component is enabled. The default is `true`.

- **android:exported**

  One of: `true`, `false`. Specifies whether this receiver is exported, i.e., whether can receive messages from other apps. The default is `false` if the element contains no intent filters, otherwise `true`.

- **android:icon**

  The resource ID of an icon for this receiver.

- **android:label**

  The string resource ID for a label.

- **android:name**

  The name of this receiver.

- **android:permission**

  A permission a broadcast sender must have to be able to send messages to this receiver.

- **android:process**

  A process name the receiver will run inside when messages arrive. If it begins with a colon (:), the process will be private to the app. If it stats with a lowercase letter, the process will be global and can be shared among apps.

It can contain the same children as subelements as an `<activity>` element, listed here:

- **intent-filter**

  See Chapter 3 for a description.

- **meta-data**

  An arbitrary name-value pair in the form `<meta-data android:name="..." android:resource="..." android:value="..." />`. You can have several of them, and they go into an `android.os.Bundle` element available as `PackageItemInfo.metaData`.

The <service> Child Element

Describes a service. For details, see Chapter 4. You can have zero, one, or more child elements of that type inside the application.

The <uses-library> Child Element

If your app needs a shared library to link against, you can specify that here. The syntax is as follows:

```
<uses-library
      android:name="string"
      android:required=["true" | "false"] />
```

Here, the `name` attribute is the name of the shared library, and by setting `required` to `true`, the app won't be installable unless the shared library requirement is met. Setting it to `false` expresses an affinity toward using that library, but your app will work without it, too.

Activity-Related Manifest Entries

In this section of the online text companion, we describe activity-related declarations inside the manifest file `AndroidManifest.xml`. Table A-3 lists all attributes of the `<activity>` element that are in the manifest file `AndroidManifest.xml`.

Table A-3. Manifest Flags for Activities

Name	Description
`android: allowEmbedded`	One of: `true`, `false`. Specifies whether the activity can be embedded inside another activity. The default is `false`.
`android: allowTaskReparenting`	One of: `true`, `false`. Specifies whether the activity can have its parent activity changed to the value given by the `taskAffinity` attribute. For example, starting a web page from an e-mail with re-parenting to the browser means the new web page gets attributed to the browser, so the next time the browser get activated, this page will be shown. This makes sense only if the `launchMode` attribute is set to `standard` or `singleTop`. The default is `false`.
`android: alwaysRetainTaskState`	One of: `true`, `false`. Only for the root activity of a task. If `true`, always return to the last state of the task, regardless of how the activity gets started. Otherwise, the Android OS might have decided after some time (say 30 minutes) that the task should be cleared, i.e., that all activities above the root activity should be removed. The default is `false`.
`android: autoRemoveFromRecents`	One of: `true`, `false`. If `false`, leave the activity on the overview screen (recents) of a task until the last activity in the task is completed. The flag overrides the activity caller's flag `FLAG_ACTIVITY_RETAIN_IN_RECENTS`. The default is `false`.
`android: banner`	For Android TV apps. Specifies the resource ID of a banner to show on the home screen. There is no default banner.
`android: clearTaskOnLaunch`	One of: `true`, `false`. Only for the root activity of a task. If `true`, returning to a task by relaunching it from the home screen will always return to the task's root activity and remove all activities on top of it. The default is `false`.
`android: colorMode`	Set this to `wideColorGamut` if you want to enable more vibrant color for devices capable of doing it. Ignored if the device cannot handle it.
`android: configChanges`	Use this to list configuration changes the activity wants to handle itself. Normally, an activity gets shut down and restarted if at runtime a configuration value changes, but if inside this list (use | as a separator), the `onConfigurationChanged()` method gets called instead. It is recommended to not use this flag. Possible list entries are shown after the table.
`android: directBootAware`	One of: `true`, `false`. Specifies whether the activity can run before the user unlocks the device. The default is `false`.
`android: documentLaunchMode`	Specifies how new instances of an activity show up in the task list (overview screen). Possible values are listed after the table.

(*continued*)

Table A-3. (*continued*)

Name	Description
`android: enabled`	One of: `true`, `false`. If `false`, the activity is disabled. The default is `true`.
`android: excludeFromRecents`	One of: `true`, `false`. If `true`, the activity will not show up on the recents (overview) screen. The default is `false`.
`android: exported`	One of: `true`, `false`. If `true`, the activity will be callable by other apps. The default is `true`.
`android: finishOnTaskLaunch`	One of: `true`, `false`. If `true`, existing instances of the same activity will be shut down when the user launches the task by choosing it from the home screen. The default is `false`. If both this setting and `allowTaskReparenting` are true, a re-parenting will not happen. Instead, the activity gets destroyed.
`android: hardwareAccelerated`	One of: `true`, `false`. If `true`, enable graphics hardware acceleration. The memory consumption might be higher. The default is `false`.
`android: icon`	The resource ID of an icon to be used for the activity, for example `@mipmap/activity1_icon`. If unset, use the app's icon instead.
`android: label`	The resource ID of a label string to be used for the activity, for example `@string/activity1_label`. If unset, use the app's label instead.
`android: launchMode`	Specifies how the activity is to be started. Details are shown in Table A-4.
`android: maxRecents`	The maximum number of activities showing up in the task's stack (overview screen). The default is 16, the minimum is 1, and the maximum is 50. If exceeded, the least recently used activity from the overview screen gets removed.
`android: maxAspectRatio`	Use this to specify the maximum aspect ratio. If exceeded on the device, the app gets letterboxed. A number greater than 1.0. On nonwearable devices, it's greater than 1.33. This is ignored if the attribute `resizeableActivity` is set to `true`.
`android: multiprocess`	One of: `true`, `false`. If `true`, the activity can be launched into the process from where the activity got instantiated. You don't normally want that. The default value is `false`.
`android: name`	The fully qualified class name of the activity. If the package name is the same as the `package` attribute of the root element, you can omit it, but add a point in front of the name then.
`android: noHistory`	One of: `true`, `false`. If the user navigates away from the activity and this flag is set to `true`, the activity will be removed from the task stack and the activity gets finished with its `finish()` method called. The default is `false`.

(*continued*)

Table A-3. (*continued*)

Name	Description
`android: parentActivityName`	Use this to specify the parent activity's name when the user navigates up by tapping on the *top* button. For API levels up to 16, also write an item like `<meta-data android:name = "android.support.PARENT_ACTIVITY" android:value = "com.example.app.MainActivity" />` inside the `<activity>` element.
`android: persistableMode`	For API levels 21 or higher, defines whether an activity gets its state persisted if the device restarts. Use `persist-RootOnly` if only the root of a task stack is to be preserved; this is also the default mode. Use `persistAcrossReboots` for activities above the root if you want their states also to be preserved. Use `persistN-ever` for no persisting. With persisting possible, you can use `onSaveInstanceState()` to do the persisting, and after reboot the `onCreate()` method gets a filled `Persistable-Bundle` object.
`android: permission`	The name of a permission the caller needs to be able to call the activity. If not set, use the `<application>` element's `permission` attribute. If both are not given, no restriction applies.
`android: process`	Normally an activity starts in the process of the app it is defined in. If you use this attribute, the process given by the value supplied here is used instead. If it starts with a `:`, a new private process gets created for that aim. If it starts with a lowercase letter, a global process with that name will be used instead.
`android: relinquishTaskIdentity`	One of: `true`, `false`. If `true`, an activity in the task stack transports its identity to the next activity on top of it once open. The default is `false`.
`android: resizeableActivity`	Added in API level 24. One of: `true`, `false`. If set to `true`, the activity can participate in a multiwindow mode, provided the device is capable of doing that. Otherwise, the activity uses the whole screen. The default value is `true`.
`android: screenOrientation`	Controls the screen orientation. Possible values are listed in Table A-5. Note that if you declare one of the portrait or landscape values there, it is considered a hard requirement of devices capable of providing that mode. So the app will not be installable on devices that cannot handle it. Consider using a `<uses-feature>` element instead; it will not prevent your app from being installed on devices with missing capabilities. See Chapter 16 for details about `<uses-feature>`.
`android: showForAllUsers`	Added in API level 23. One of: `true`, `false`. If `true`, also show the activity to other users (those who haven't started it).

(*continued*)

Table A-3. (*continued*)

Name	Description
`android: stateNotNeeded`	One of: `true`, `false`. If `true`, you express that in the process of killing and restarting the activity a saving and restoring of the activity's state is not necessary. The method `OnSaveInstanceState()` is then possibly not called, and the `onCreate()` method will get a `null`-valued bundle as an argument. The default is `false`.
`android: supportsPictureInPicture`	Added in API level 24. One of: `true`, `false`. If `true`, a picture-in-picture display for Android TV will be possibly available to the activity. You then also must make sure the attribute `resizeableActivity` is set to `true`.
`android: taskAffinity`	Normally all activities of an app have the same affinity, which means they all go to the same task stack once active. Their affinity is then determined by the root activity's name. If you set an activity's `taskAffinity`, though, you can express the activity as belonging to another task stack, which takes effect once the activity is re-parented or launched with the `FLAG_ACTIVITY_NEW_TASK` flag set. If unset, take the `application` element's affinity, and if that one is unset, too, take the package name.
`android: theme`	Points to a style resource to be used as the activity's style theme. If unset, take the `application` element's theme, and if this is unset, too, take the default system theme.
`android: uiOptions`	Extra UI options to use. One of: `none` for nothing extra (the default), `splitActionBarWhenNarrow` to have an additional action bar at the bottom of the screen when horizontal space gets scarce.
`android: windowSoftInputMode`	Specifies how the soft-keyboard interacts with the activity. See Table A-6 for details. You can combine a `state` and an `adjust` setting by using a | separator if necessary.

For `android:configChanges` to be forwarded to the `Activity` class, you can supply a list of the following values, separated by bars (|):

- `density` for the display density
- `fontScale` for the font scaling factor
- `keyboard` for the keyboard type
- `keyboardHidden` for the keyboard accessibility
- `layoutDirection` if changing from left to right to right to left, or the other way round (only starting at API level 17)
- `locale` for the locale
- `mcc` for the mobile country code

- `mnc` for the mobile network code
- `navigation` for the navigation type
- `orientation` for the screen orientation
- `screenLayout` for the screen layout
- `screenSize` for the screen size
- `smallestScreenSize` for the physical screen size
- `touchscreen` if the touchscreen has changed
- `uiMode` if the user interface has changed, for example if the device was placed in a dock

For the `documentLaunchMode` flag to control how to handle new document-related activities, you can provide one of the following:

- `intoExisting` to search for a matching (`ComponentName` and data URI) task and possibly restart it with the root activity receiving `onNewIntent()`) calls
- `always` for the activity to always create a new task for a document
- `none` (this is the default value) to not create new tasks unless `FLAG_ACTIVITY_NEW_TASK` is set by the caller
- `never` to never create a new task

If other than `none` and `never`, `launchMode` must have been set to `standard` for the setting to take effect.

***Table A-4.** Launch Modes by launchMode Setting*

Name	Description
`standard`	The default mode. A `standard` activity can be instantiated many times. Every time a new instance is created, it will show up as a new entry in the task list.
`singleTop`	Same as `standard`, but with the difference that if an instance of the activity exists and is on top of the task's activity stack, this existing activity's `onNewIntent()` method gets called instead.
`singleTask`	There can be only one instance of this activity in the task's stack. New activity launches of the same activity type will be routed to the `onNewIntent()` method of the existing activity. Other activities instantiated will pile up normally on top of this activity. Consider this for exceptional purposes only.
`singleInstance`	Same as `singleTask`, except that other activities will not pile up here. So the task's stack will only ever contain one element. Consider this for exceptional purposes only.

Table A-5. Screen Orientation Modes

Name	Description
`unspecified`	The default value. Here the Android OS chooses the orientation.
`behind`	Uses the same orientation as the activity underneath in the task stack.
`landscape`	Landscape orientation.
`portrait`	Portrait orientation.
`reverseLandscape`	Landscape orientation but flipped.
`reversePortrait`	Portrait orientation but flipped.
`sensorLandscape`	Like `landscape` but can be flipped based on the device sensor.
`sensorPortrait`	Like `portrait` but can be flipped based on the device sensor.
`userLandscape`	Like `sensorLandscape` but also influenced by the user preferences.
`userPortrait`	Like `sensorPortrait`, but also influenced by the user preferences.
`sensor`	The orientation will be based on the device's orientation sensor.
`fullSensor`	Like `sensor`, but also allows flipping.
`nosensor`	The orientation sensor will be disabled.
`user`	The orientation is based on the user preferences.
`fullUser`	Added in API level 18. Uses user preferences if the user has locked sensor-based rotation. Otherwise, same as `fullSensor`.
`locked`	Locks to the current rotation.

Table A-6. Soft-Keyboard Input Modes

Name	Description
`stateUnspecified`	The default setting. The Android OS decides how to present the soft-keyboard.
`stateUnchanged`	Use the last state.
`stateHidden`	Hide the soft keyboard for forward navigation (the user affirmatively chooses the activity).
`stateAlwaysHidden`	Hide the soft keyboard when the activity has focus.
`stateVisible`	Show the soft keyboard for forward navigation (the user affirmatively chooses the activity).
`stateAlwaysVisible`	Show the soft keyboard when the activity has focus.
`adjustUnspecified`	The default adjustment. Android decides whether the activity's layout needs an adjustment to make space for the soft keyboard. The latter depends on the scrolling capability of the layout.
`adjustResize`	Forces resizing the activity's layout to make space for the soft keyboard.
`adjustPan`	Does not force a resizing of the layout; instead, reduces its space and possibly pans the view on the layout to make sure the input widget is visible.

Intent Constituent Parts

This section of the online text companion describes various elements of intents. Intents get used inside `<intent-filter>` elements in the manifest file. For details, see Chapter 3.

Intent Actions

These are generic action attributes to be used in `<action>` elements inside intent filters, and they are specified by the constants with names `ACTION_*` inside class `android.content.Intent`, as shown in Tables A-7 and A-8.

***Table A-7.** Intent Actions for Activities*

Action	Description
`ALL_APPS`	Lists all available applications.
`ANSWER`	Handles an incoming phone call.
`APPLICATION_PREFERENCES`	An activity that provides a user interface for adjusting application preferences.
`APP_ERROR`	Indicates the user clicked the Report button in the crash/ANR dialog.
`ASSIST`	Performs assist action.
`ATTACH_DATA`	Used to indicate that some piece of data should be attached to some other place.
`BUG_REPORT`	Shows activity for reporting a bug.
`CALL`	Performs a call to someone specified by the data.
`CALL_BUTTON`	The user pressed the “call” button to go to the dialer or other appropriate UI for placing a call.
`CARRIER_SETUP`	Main entry point for carrier setup apps.
`CHOOSER`	Displays an activity chooser, allowing the user to pick what they want to before proceeding.
`CREATE_DOCUMENT`	Allows the user to create a new document.
`CREATE_SHORTCUT`	Creates a shortcut.
`DEFAULT`	Synonym for `VIEW`; the “standard” action that is performed on a piece of data.
`DELETE`	Deletes the given data from its container.
`DIAL`	Dials a number as specified by the data.
`EDIT`	Provides explicit editable access to the given data.
`FACTORY_TEST`	Main entry point for factory tests.
`GET_CONTENT`	Allows the user to select a particular kind of data and return it.
`INSERT`	Inserts an empty item into the given container.

(*continued*)

Table A-7. (*continued*)

Action	Description
INSERT_OR_EDIT	Picks an existing item, or inserts a new item, and then edits it.
INSTALL_FAILURE	Activity to handle split installation failures.
INSTALL_PACKAGE	Launches application installer.
MAIN	Starts as a main entry point; does not expect to receive data.
MANAGE_NETWORK_USAGE	Shows settings for managing network data usage of a specific application.
OPEN_DOCUMENT	Allows the user to select and return one or more existing documents.
OPEN_DOCUMENT_TREE	Allows the user to pick a directory subtree.
PASTE	Creates a new item in the given container, initializing it from the current contents of the clipboard.
PICK	Picks an item from the data, returning what was selected.
PICK_ACTIVITY	Picks an activity given an intent, returning the class selected.
POWER_USAGE_SUMMARY	Shows power usage information to the user.
PROCESS_TEXT	Processes a piece of text.
QUICK_VIEW	Quick view the data.
RUN	Runs the data, whatever that means.
SEARCH	Performs a search.
SEARCH_LONG_PRESS	Starts action associated with long-pressing the search key.
SEND	Delivers some data to someone else.
SENDTO	Sends a message to someone specified by the data.
SEND_MULTIPLE	Delivers multiple data to someone else.
SET_WALLPAPER	Shows settings for choosing wallpaper.
SHOW_APP_INFO	Launches an activity showing the app information.
SYNC	Performs a data synchronization.
SYSTEM_TUTORIAL	Starts the platform-defined tutorial. The extra datum `SearchManager.QUERY` (a string) is the text to search for.
UNINSTALL_PACKAGE	Launches application uninstaller.
VIEW	Displays the data to the user.
VOICE_COMMAND	Starts Voice Command.

Table A-8. Intent Actions for Broadcasts

Action	Description
AIRPLANE_MODE_CHANGED	Toggles the device's Airplane mode.
APPLICATION_RESTRICTIONS_CHANGED	Application restrictions were changed.
BATTERY_CHANGED	A broadcast containing the charging state, level, and other information about the battery.
BATTERY_LOW	A low battery condition occurs on the device.
BATTERY_OKAY	The battery is now okay after being low.
BOOT_COMPLETED	The device has finished booting.
CAMERA_BUTTON	The camera button was pressed.
CLOSE_SYSTEM_DIALOGS	Closing a temporary system dialog is pending.
CONFIGURATION_CHANGED	The current device configuration has changed, for example the orientation or locale.
DATE_CHANGED	The date has changed.
DEVICE_STORAGE_LOW	Defunct starting with API level 26. Apps should instead use `getCacheDir()` so the system can free up storage when needed.
DEVICE_STORAGE_OK	Defunct starting with API level 26. Apps should instead use `getCacheDir()` so the system can free up storage when needed.
DOCK_EVENT	A change in the physical docking state of the device occurred.
DREAMING_STARTED	The system started dreaming.
DREAMING_STOPPED	The system stopped dreaming.
EXTERNAL_APPLICATIONS_AVAILABLE	Resources for a set of packages (which were previously unavailable) are currently available since the media on which they exist is available.
EXTERNAL_APPLICATIONS_UNAVAILABLE	The media on which external apps exist is unavailable.
GET_RESTRICTION_ENTRIES	The system tells an app to query restrictions that are to be imposed on restricted users.
GTALK_SERVICE_CONNECTED	A GTalk connection has been established.
GTALK_SERVICE_DISCONNECTED	A GTalk connection has been disconnected.
HEADSET_PLUG	A wired headset has been plugged in or unplugged.
INPUT_METHOD_CHANGED	An input method has been changed.
LOCALE_CHANGED	The current device's locale has changed.
LOCKED_BOOT_COMPLETED	The device has finished booting but is still in the "locked" state.
MANAGED_PROFILE_ADDED	A broadcast sent to the primary user when an associated managed profile is added.

(*continued*)

Table A-8. (*continued*)

Action	Description
MANAGED_PROFILE_AVAILABLE	A broadcast sent to the primary user when an associated managed profile has become available.
MANAGED_PROFILE_REMOVED	A broadcast sent to the primary user when an associated managed profile is removed.
MANAGED_PROFILE_UNAVAILABLE	A broadcast sent to the primary user when an associated managed profile has become unavailable.
MANAGED_PROFILE_UNLOCKED	A broadcast sent to the primary user when the credential-encrypted private storage for an associated managed profile is unlocked.
MANAGE_PACKAGE_STORAGE	The package management should be started after the user acknowledges a low memory condition.
MEDIA_BAD_REMOVAL	An external medium was removed from the SD card slot without properly being unmounted.
MEDIA_BUTTON	The media button was pressed.
MEDIA_CHECKING	An external medium is present and being disk-checked. The path to the mount point of the checking media is contained in the `Intent.mData` field.
MEDIA_EJECT	The user has expressed the desire to remove the external storage medium.
MEDIA_MOUNTED	An external medium is present and mounted.
MEDIA_NOFS	An external medium is present, but it is using an incompatible file system. The path to the mount point is contained in the `Intent.mData` field.
MEDIA_REMOVED	An external medium has been removed.
MEDIA_SCANNER_FINISHED	The media scanner has finished scanning a directory.
MEDIA_SCANNER_SCAN_FILE	Request the media scanner to scan a file and add it to the media database.
MEDIA_SCANNER_STARTED	The media scanner has started scanning a directory.
MEDIA_SHARED	The external medium is unmounted because it is being shared as an USB mass storage.
MEDIA_UNMOUNTABLE	The external medium is present but cannot be mounted.
MEDIA_UNMOUNTED	An external medium is present but not mounted.
MY_PACKAGE_REPLACED	A new version of your app has been installed, replacing an existing one.
NEW_OUTGOING_CALL	An outgoing call is about to be placed.
PACKAGES_SUSPENDED	Packages have been suspended.
PACKAGES_UNSUSPENDED	Packages have been unsuspended.
PACKAGE_ADDED	A new application package has been installed on the device.

(*continued*)

Table A-8. (*continued*)

Action	Description
PACKAGE_CHANGED	An existing app package has been changed; for example, it was enabled or disabled.
PACKAGE_DATA_CLEARED	The user has cleared the data of a package.
PACKAGE_FIRST_LAUNCH	Sent to the installer package of an application when that app is first launched.
PACKAGE_FULLY_REMOVED	An app has been completely removed from the device.
PACKAGE_INSTALL	Unused and deprecated since API level 14.
PACKAGE_NEEDS_VERIFICATION	Sent to the system package verifier when a package needs to be verified.
PACKAGE_REMOVED	An app has been removed from the device.
PACKAGE_REPLACED	A new version of an app has been installed, replacing an existing version.
PACKAGE_RESTARTED	The user has restarted a package after all of its processes have been killed.
PACKAGE_VERIFIED	Sent to the system package verifier when a package is verified.
POWER_CONNECTED	External power has been connected to the device.
POWER_DISCONNECTED	External power has been removed from the device.
PROVIDER_CHANGED	Some content providers have parts of their namespace changed.
QUICK_CLOCK	Sent when the user taps the clock widget in the system's "quick settings" area.
REBOOT	Reboots the device.
SCREEN_OFF	Sent when the device goes to sleep and becomes noninteractive.
SCREEN_ON	Sent when the device wakes up and becomes interactive.
SHUTDOWN	The device is shutting down.
TIMEZONE_CHANGED	The time zone has changed.
TIME_CHANGED	The time was set.
TIME_TICK	The current time has changed.
UID_REMOVED	A user ID has been removed from the system.
UMS_CONNECTED	Deprecated in API level 14. It has been replaced by `android.os.storage.StorageEventListener`.
UMS_DISCONNECTED	Deprecated in API level 14. It has been replaced by `android.os.storage.StorageEventListener`.
USER_BACKGROUND	A user switch is happening, causing the process's user to be sent to the background.

(*continued*)

Table A-8. (*continued*)

Action	Description
USER_FOREGROUND	A user switch is happening, causing the process's user to be brought to the foreground.
USER_INITIALIZE	Sent the first time a user is starting.
USER_PRESENT	Sent when the user is present after the device wakes up.
USER_UNLOCKED	Sent when the credential-encrypted private storage has become unlocked for the target user.
WALLPAPER_CHANGED	Deprecated in API level 16. Modern apps should instead use WindowManager.Layout-Params.FLAG_SHOW_WALLPAPER to have the wallpaper shown behind their UI.

To use generic actions for the name attribute of the <action> element, prepend android.intent.action. to the action name used from the table. For actions you defined yourself, prepend your package name.

Intent Categories

Generic category attributes to be used in <action> elements inside intent filters are specified by the constants with names CATEGORY_* inside class android.content.Intent, as shown in Table A-9.

Table A-9. Intent Categories

Category	Description
ALTERNATIVE	Set if the activity should be considered as an alternative action to the data the user is currently viewing.
APP_BROWSER	Used with ACTION_MAIN to launch the browser application.
APP_CALCULATOR	Used with ACTION_MAIN to launch the calculator application.
APP_CALENDAR	Used with ACTION_MAIN to launch the calendar application.
APP_CONTACTS	Used with ACTION_MAIN to launch the contacts application.
APP_EMAIL	Used with ACTION_MAIN to launch the e-mail application.
APP_GALLERY	Used with ACTION_MAIN to launch the gallery application.
APP_MAPS	Used with ACTION_MAIN to launch the maps application.
APP_MARKET	This activity allows the user to browse and download new applications.
APP_MESSAGING	Used with ACTION_MAIN to launch the messaging application.
APP_MUSIC	Used with ACTION_MAIN to launch the music application.
BROWSABLE	Activities that can be safely invoked from a browser must support this category.

(*continued*)

Table A-9. (*continued*)

Category	Description
CAR_DOCK	An activity to run when device is inserted into a car dock.
CAR_MODE	Used to indicate that the activity can be used in a car environment.
DEFAULT	Set if the activity should be an option for the default action (center press) to perform on a piece of data.
DESK_DOCK	An activity to run when device is inserted into a car dock.
DEVELOPMENT_PREFERENCE	This activity is a development preference panel.
EMBED	Capable of running inside a parent activity container.
FRAMEWORK_ INSTRUMENTATION_TEST	To be used as code under test for framework instrumentation tests.
HE_DESK_DOCK	An activity to run when device is inserted into a digital (high-end) dock.
HOME	This is the home activity, which is the first activity that is displayed when the device boots.
INFO	Provides information about the package it is in; typically used if a package does not contain a LAUNCHER to provide a front door to the user without having to be shown in the all apps list.
LAUNCHER	Should be displayed in the top-level launcher.
LEANBACK_LAUNCHER	Indicates an activity optimized for Leanback mode and that it should be displayed in the Leanback launcher.
LE_DESK_DOCK	An activity to run when the device is inserted into a analog (low-end) dock.
MONKEY	This activity may be exercised by the monkey or other automated test tools.
OPENABLE	Used to indicate that an intent only wants URIs that can be opened with openFileDescriptor(Uri, String).
PREFERENCE	This activity is a preference panel.
SAMPLE_CODE	To be used as a code example (i.e., not part of the normal user experience).
SELECTED_ALTERNATIVE	Set if the activity should be considered as an alternative selection action to the data the user has currently selected.
TAB	Intended to be used as a tab inside a containing TabActivity.
TEST	To be used as a test (i.e., not part of the normal user experience).
TYPED_OPENABLE	Used to indicate that an intent filter can accept files that are not necessarily openable by openFileDescriptor(Uri, String) but at least streamable via openTypedAsset- FileDescriptor(Uri, String, Bundle) using one of the stream types exposed via getStream- Types(Uri, String).
UNIT_TEST	To be used as a unit test (e.g., run through the test harness).
VOICE	Categories for activities that can participate in voice interaction.
VR_HOME	An activity to use for the launcher when the device is placed in a VR headset viewer.

To use one of the categories from the table for the `name` attribute of the `<category>` element, prepend `android.intent.category.` to the category name from the table. For custom categories you defined yourself, prepend your package name.

Intent Extra Data

Note that the names in the following list are constant names from the `android.content.Intent` class, so for example write `Intent.EXTRA_ALARM_COUNT` as a key.

- **EXTRA_ALARM_COUNT**

  With the AlarmManager (`android.app.AlarmManager`) sending alarm events to registered broadcast receivers, this is an extra field that tells about the number of current and pending alarm events. Pending alarm events are events that, for example, couldn't be sent because the device was down.

- **EXTRA_BCC**

  When calling an e-mailer, a `String[]` array holding e-mail addresses that should be blind carbon copied.

- **EXTRA_CC**

  When calling an e-mailer, a `String[]` array holding e-mail addresses that should be carbon copied.

- **EXTRA_CHANGED_COMPONENT_NAME**

  This constant was deprecated in API level 7. Do not use it (instead use `EXTRA_CHANGED_COMPONENT_NAME_LIST`).

- **EXTRA_CHANGED_COMPONENT_NAME_LIST**

  An array of the components that have changed. Contains component class names or package names (treating the packages as super-components).

- **EXTRA_DATA_REMOVED**

  In `ACTION_PACKAGE_REMOVED` intents, this boolean extra field is set to `true` if the data is being removed as well.

- **EXTRA_DOCK_STATE**

  An `int` extra field in `ACTION_DOCK_EVENT` intents telling about the dock state. Possible values are as follows:

  - **EXTRA_DOCK_STATE_HE_DESK**

    The device is in a digital (high-end) dock.

  - **EXTRA_DOCK_STATE_LE_DESK**

    The device is in an analog (low-end) dock.

  - **EXTRA_DOCK_STATE_CAR**

    The device is in a car dock.

- **EXTRA_DOCK_STATE_DESK**

  The device is in a desk dock.

- **EXTRA_DOCK_STATE_UNDOCKED**

  The device is not in any dock.

- **EXTRA_DONT_KILL_APP**

  For the action intents `ACTION_PACKAGE_REMOVED` and `ACTION_PACKAGE_CHANGED`, setting this to `true` overrides the default action and hence signals the application is not to be restarted.

- **EXTRA_EMAIL**

  When calling an e-mailer, a `String[]` array holding recipient e-mail addresses.

- **EXTRA_INITIAL_INTENTS**

  With an action intent `ACTION_CHOOSER`, a `Parcelable[]` array of intent or `LabeledIntent` objects of additional activities to place at the front of the list of choices presented to the user.

- **EXTRA_INTENT**

  For `ACTION_PICK_ACTIVITY` or `ACTION_CHOOSER` activities, this field must contain the string representation of the target intent being issued.

- **EXTRA_KEY_EVENT**

  For intents triggered by key events, this is the corresponding `KeyEvent` object.

- **EXTRA_ORIGINATING_URI**

  For actions `ACTION_INSTALL_PACKAGE` and `ACTION_VIEW`, this contains the originating URI of the installed APK.

- **EXTRA_PHONE_NUMBER**

  This is a string holding the phone number originally entered in action `ACTION_NEW_OUTGOING_CALL`, or the actual number to call for an `ACTION_CALL` action.

- **EXTRA_REFERRER**

  A general-purpose field supplying information about who is launching an activity. This field contains a `Uri` object (scheme `http:` or `https:`, or also `android-app:` for native applications). However, to retrieve this value in a client, use `getReferrer()` instead. It is also valid for applications to instead supply `EXTRA_REFERRER_NAME` for cases where they can create only a string, not a URI (the field here, if supplied, will take precedence, though).

- **EXTRA_REFERRER_NAME**

  A referrer as a string. See `EXTRA_REFERRER`.

- **EXTRA_REMOTE_INTENT_TOKEN**

  A general-purpose field for remote intents.

- **EXTRA_REPLACING**

  If this is set to `true`, the broadcast will immediately be followed by an "add" broadcast for a different version of the same package.

- **EXTRA_SHORTCUT_ICON**

  Deprecated in API level 26. Defines an icon of a shortcut. Instead, use `createShortcutResultIntent(ShortcutInfo)`.

- **EXTRA_SHORTCUT_ICON_RESOURCE**

  Deprecated in API level 26. Defines the resource ID of a shortcut icon. Replaced with `createShortcutResultIntent(ShortcutInfo)`.

- **EXTRA_SHORTCUT_INTENT**

  Deprecated in API level 26. Defines the intent of a shortcut. Replaced with `createShortcutResultIntent(ShortcutInfo)`.

- **EXTRA_STREAM**

  A `content:` URI holding a stream of data associated with the intent, used with `ACTION_SEND` to supply the data being sent.

- **EXTRA_SHORTCUT_NAME**

  Deprecated in API level 26. Defines the shortcut name. Replaced with `cr eateShortcutResultIntent(ShortcutInfo)`.

- **EXTRA_SUBJECT**

  For e-mailers, a string holding the desired subject line of a message.

- **EXTRA_TEMPLATE**

  For content providers, the initial data to place in a newly created record. Use with `ACTION_INSERT`. The data here is a `Map` containing the same fields as would be given to the underlying `ContentProvider.insert()` call.

- **EXTRA_TEXT**

  This is a `CharSequence` associated with the intent, used with `ACTION_SEND` to supply the literal data to be sent. Note that this may be a styled `CharSequence`, so you must use `Bundle.getCharSequence()` to retrieve it.

- **EXTRA_TITLE**

  A `CharSequence` dialog title to provide to the user when used with a `ACTION_CHOOSER`.

- **EXTRA_UID**

  Used as an `int` extra field in `ACTION_UID_REMOVED` intents to supply the UID the package had been assigned to. This is also an optional extra value in `ACTION_PACKAGE_REMOVED` or `ACTION_PACKAGE_CHANGED` for the same purpose.

Intent Flags

Tables A-10, A-11, and A-12 show the details for intent flags you can set inside your code.

***Table A-10.** Activity Intent-Related Flags*

Name	Description
`FLAG_ACTIVITY_BROUGHT_TO_FRONT`	This flag is set by the system and corresponds to the `singleTask` mode, with only one instance of the activity in the task's stack. New activity launches of the same activity type will be routed to the `onNewIntent()` method of the existing activity. Other activities instantiated will pile up normally on top of this activity.
`FLAG_ACTIVITY_CLEAR_TASK`	This flag will cause any existing task that would be associated with the activity to be cleared before the activity is started. That is, the activity becomes the new root of an otherwise empty task, and any old activities are finished. This can be used in conjunction with `FLAG_ACTIVITY_NEW_TASK`.
`FLAG_ACTIVITY_CLEAR_TOP`	If set and the activity being launched is already running in the current task, all of the other activities on top of it will be closed, and this intent will be delivered by `onNewIntent()` to the (now on top) old activity as a new intent.
`FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS`	If set, the new activity is not kept in the list of recently launched activities.
`FLAG_ACTIVITY_FORWARD_RESULT`	Use this to have the reply target from an existing instance of the new activity forwarded to the newly built activity. This way the new activity can call `setResult(int)` and have that result sent back to the reply target of the original activity.
`FLAG_ACTIVITY_LAUNCH_ADJACENT`	Used only in split-screen multiwindow mode and only in conjunction with `FLAG_ACTIVITY_NEW_TASK`. The new activity will be displayed adjacent to the one launching it. Note that setting `FLAG_ACTIVITY_MULTIPLE_TASK` is required if you want a new instance of an existing activity to be created.
`FLAG_ACTIVITY_LAUNCHED_FROM_HISTORY`	Indicates that this activity is being launched from history (long-press the home key).
`FLAG_ACTIVITY_NEW_DOCUMENT`	This flag is used to open a document into a new task rooted at and on top of the activity launched by this intent. If this flag is being used to create a new recents entry, then by default that entry will be removed once the activity is finished.
`FLAG_ACTIVITY_NO_ANIMATION`	This flag will prevent the system from applying an activity transition animation while going to the next activity state. An animation still can occur if the currently displayed activity has animation enabled.

(*continued*)

Table A-10. (*continued*)

Name	Description
FLAG_ACTIVITY_NO_USER_ACTION	This flag will prevent the normal onUserLeave- Hint() callback from occurring on the current frontmost activity before it is paused as the newly started activity is brought to the front. If an activity is ever started via any non-user-driven events such as a phone call receipt or an alarm handler, this flag should be passed to Context.startActivity(), ensuring that the pausing activity does not think the user has acknowledged its notification.
FLAG_ACTIVITY_PREVIOUS_IS_TOP	If set, the current activity will not be counted as the top activity for deciding whether the new intent should be delivered to the top instead of starting a new instance. Instead, the second to last activity will be used as the top.
FLAG_ACTIVITY_REORDER_TO_FRONT	Specify this if you want a previously existing activity of the same type to be brought to front first. Say you have a stack A, B, C, D. Now with this flag set and the intent addressing B, the stack will become A, C, D, B. This flag has no effect if also FLAG_ACTIVITY_CLEAR_TOP is given.
FLAG_ACTIVITY_RETAIN_IN_RECENTS	Usually a document created by FLAG_ACTIVITY_NEW_DOCUMENT will be removed from the recents list if the document gets closed. Use this flag if instead you want the document to retain in the recents list.
FLAG_ACTIVITY_TASK_ON_HOME	This flag, only to be used in conjunction with FLAG_ACTIVITY_NEW_TASK, will cause a newly launching task to be placed on top of the current home activity task. This means pressing "back" from the task will always return the user to home.

Table A-11. Broadcast Intent-Related Flags

Name	Description
FLAG_RECEIVER_FOREGROUND	Set this if you want the recipient of a broadcast to be allowed to run at foreground priority (a shorter timeout interval applies). Not using this flag, during normal broadcasts the receivers are not automatically hoisted out of the background priority class.
FLAG_RECEIVER_NO_ABORT	If this is an ordered broadcast, don't allow receivers to abort the broadcast. They can still propagate results through to later receivers, but they cannot prevent later receivers from seeing the broadcast.
FLAG_RECEIVER_REGISTERED_ONLY	Setting this to true will cause registered receivers to be called only if a broadcast happens. No BroadcastReceiver components will be launched.
FLAG_RECEIVER_REPLACE_PENDING	If set, when sending a broadcast, the new broadcast will replace any existing pending broadcast that matches it (defined by Intent.filterEquals(...) returning true).
FLAG_RECEIVER_VISIBLE_TO_INSTANT_APPS	If set, the broadcast will be visible to receivers in instant apps. By default, instant apps will not receive broadcasts (see Chapter 17). This flag has no effect when used by an instant app as a broadcast emitter.

Table A-12. Other Intent Flags

Name	Description
FLAG_DEBUG_LOG_RESOLUTION	This comes handy for debugging. When set, log messages will be printed during the resolution of this intent to show you what has been found to create the final resolved list.
FLAG_EXCLUDE_STOPPED_PACKAGES	If you want to disregard stopped packages while trying to match intents, set this flag. Note that this is not the expected behavior of apps, but you can use it for corner cases.
FLAG_FROM_BACKGROUND	Use this to indicate the caller is running in background mode and is thus not part of a direct user interaction. Since background operations are more likely to be killed by the Android OS, checking this flag might be important for mission-critical broadcast receivers.
FLAG_GRANT_PERSISTABLE_URI_PERMISSION	In conjunction with FLAG_GRANT_READ_URI_PERMISSION and/or FLAG_GRANT_WRITE_URI_PERMISSION, setting this flag indicates the URI permission grant to be persistable across device reboots until explicitly revoked with revokeUriPermission(Uri, Int). This flag only offers the grant for possible persisting; the receiving application must still call takePersistableUriPermission(Uri, Int) to actually persist.
FLAG_GRANT_PREFIX_URI_PERMISSION	In conjunction with FLAG_GRANT_READ_URI_PERMISSION and/or FLAG_GRANT_WRITE_URI_PERMISSION, setting this flag indicates that the URI permission grant applies to any URI that is a prefix match against the original granted URI. (Without this flag, the URI must match exactly for access to be granted.) Another URI is considered a prefix match only when the scheme, authority, and all path segments defined by the prefix are an exact match.
FLAG_GRANT_READ_URI_PERMISSION	Use this if you want the recipient of this intent to be granted permission to perform read operations on the URI in the intent's data and any URIs specified in its ClipData.
FLAG_GRANT_WRITE_URI_PERMISSION	Use this if you want the recipient of this intent to be granted permission to perform write operations on the URI in the intent's data and any URIs specified in its ClipData.
FLAG_INCLUDE_STOPPED_PACKAGES	Use this to override the flag FLAG_EXCLUDE_STOPPED_PACKAGES. If the flag FLAG_EXCLUDE_STOPPED_PACKAGES is not given, you don't have to do anything since including stopped packages is the default.

The Service Class

The `android.app.Service` class methods get listed and described later in the chapter in Table A-13.

***Table A-13.** The android.app.Service Class*

Method	Description
All `Context` methods	Since the `Service` class inherits from `android.content.Context`, it inherits all its methods and attributes. See the API reference.
`getApplication():Application`	Returns the application that owns this service.
`onBind(intent:Intent):IBinder`	Use this to communicate with the service. The `android.os.IBinder` interface you receive in that method contains a couple of methods to check for the connection sanity; use them excessively to check whether connections are still alive!
`onConfigurationChanged(newConfig: Configuration): Unit`	Called by the system when the device configuration changes while your component is running.
`onCreate():Unit`	Called by the system when the service is first created.
`onDestroy():Unit`	Called by the system to notify a service that it is no longer used and is being removed. This happens both for explicitly started services not stopped by clients or by themselves and for bound services if all bindings were released.
`onLowMemory():Unit`	This is called when the overall system is running low on memory, and actively running processes should trim their memory usage.
`onRebind(intent:Intent):Unit`	Called when new clients have connected to the service, after it had previously been notified that all clients had disconnected in its `onUnbind(Intent)`.
`onStart(intent:Intent, startId:Int): Unit`	Deprecated. Instead, implement `onStartCommand(Intent, Int, Int)`.
`onStartCommand(intent:Intent, flags:Int, startId:Int):Int`	Called by the system every time a client explicitly starts the service by calling `startService(Intent)`, providing the arguments it supplied and a unique integer token representing the start request. Note that automatic starts triggered by binding services do not yield this method being called!
`onTaskRemoved(Intent rootIntent:Intent):Unit`	This is called if the service is currently running and the user has removed a task that comes from the service's application.
`onTrimMemory(level:Int):Unit`	Called when the operating system has determined that it is a good time for a service to release unneeded memory from its process.

(*continued*)

Table A-13. (*continued*)

Method	Description
`onUnbind(intent:Intent): Boolean`	Called when all clients have abandoned their binding to the service. The argument contains the intent when `bindService()` was called, but without any extra data added to the intent object. Let it return `true` if you want later binding attempts from clients to automatically invoke the `onRebind()` method of the service.
`startForeground(id:Int, notification:Notification):Unit`	If your service is started via `startService(Intent)` or `startForegroundService(Intent)`, then also make this service (actually) run in the foreground, supplying the ongoing notification to be shown to the user while in this state. Since Android 8.0 (API level 26), foreground services *must* call this to really run in the foreground, and skipping this call for services started via `startForegroundService(Intent)` will lead to an error.
`stopForeground(flags:Int):Unit`	Remove this service from foreground state, allowing it to be killed if more memory is needed.
`stopForeground( removeNotification: Boolean): Unit`	Same as `stopForeground(Int)` with `STOP_FOREGROUND_REMOVE` flag set. This means any status bar notification registered will be removed as well.
`stopSelf():Unit`	Stop the service, if it was previously started. This way, a service can stop itself.
`stopSelf(startId:Int):Unit`	Old version of `stopSelfResult(int)` that doesn't return a result.
`stopSelfResult(startId:Int): Boolean`	With the `onStartCommand()` callback, you receive an integer ID tied to a service startup. If you use this very same ID as an argument to the `stopSelfResult()` method, a stopping of the service will actually not occur if some other component meanwhile started the service (ignorant of whether the service is already running or not, which is allowed). So, the stopping by this method happens only when it matches the most recent start. Returns `true` if the service is going to actually be stopped.
`protected dump(fd:FileDescriptor, writer:PrintWriter, args:Array<String>)`	Gets called when you invoke `adb shell dumpsys activity service <yourservicename>` in a shell and the service in question is running. Use this for diagnostic purposes during development, and write the diagnostic information to the `PrintWriter` argument. You don't have to close the `PrintWriter` argument after the dump; Android does it for you.

The service invocation methods all live in the `android.content.Context` class. We describe them in Table A-14.

***Table A-14.** Service Invocation Methods*

Method	Description
`bindService(service: Intent, conn: ServiceConnection, flags: Int): Boolean`	Binds (connects) to a service. Binding was described in Chapter X.
`startForegroundService(service: Intent): ComponentName`	Starts a service with the implicit promise that the service will call `startForeground(Int, Notification)` once it begins running. Returns the service's name if the service is running after this call (whether it was newly started or was already running when the call occurred) or `null` if the service doesn't exist.
`startService(service: Intent): ComponentName`	Starts a service if it is not already running. Returns the service's name if the service is running after this call (whether it was newly started or was already running when the call occurred) or `null` if the service doesn't exist.
`stopService(service: Intent): Boolean`	Requests that a given service be stopped.
`unbindService(conn: ServiceConnection): Unit`	Disconnect from a service. This doesn't automatically mean the service will be stopped; it just tells that this service client's binding is released.

Cursor Interface

The `android.database.Cursor` interface declares various methods for data retrieval from a database or content provider.

Administrative Methods

- **override fun close():Unit**

  Use this to close resources. The cursor object is not supposed to be usable any longer after this method call.

- **override fun isClosed(): Boolean**

  Return `true` if the cursor is closed and possibly unusable.

- **override fun deactivate(): Unit**

  This is deprecated; don't use it. Just make this a no-op. There is no need to do a deactivation because a reactivation is deprecated, too. Clients should invoke `close()` instead. If you use it nevertheless, the cursor is supposed to be deactivated and possibly unusable after this call, maybe to temporarily free resources. A later `requery()` must reactivate the cursor.

- **override fun requery(): Boolean**

  This is deprecated; don't use it. Make this a no-op and let it return `true`. Clients can instead just request a new cursor. If you use it nevertheless, let it perform a requery of the same request that created the cursor object, possibly refreshing it. If the cursor was deactivated, it must be reactivated after this call. This is supposed to return `true` if the requery was successful, otherwise `false`. If it returns `false`, the cursor may be invalid; i.e., clients must not rely on its validity.

- **override fun setExtras(extras: Bundle?):Unit**

  Use this from inside your content provider if you want to provide any out-of-band values to clients. You should not normally use this for functional purposes because it runs out of the contract and thus has a bad smell. You may use it for nonfunctional purposes, though, for example to process numbers when the cursor cannot yet provide real values but a `requery()` is required to try again.

- **override fun getExtras(): Bundle**

  Clients may use this to get out-of-band data provided by your content provider. It must not return `null`; for empty sets, return `Bundle.EMPTY` instead.

- **override fun respond(extras: Bundle?): Bundle**

  Clients may use this to communicate with the cursor in an out-of-band manner. For example, they may query the cursor state if the cursor does not yet provide actual values but a `requery()` is needed. You should not normally use this for functional purposes, since this is the responsibility of the `ContentProvider` methods. It must not return `null` because empty sets return `Bundle.EMPTY` instead. Other than suggested by the parameter signature, the `bundle` parameter should not be `null` by API contract, but surely it won't hurt to check that anyway.

Navigation

- **override fun getCount(): Int**

  Returns the numbers of rows in the cursor.

- **override fun moveToPosition(position: Int): Boolean**

  Moves the cursor to an absolute position. `-1` is *before* the first row (you cannot retrieve columns there), `0` is the first row, `count-1` is the last row, and `count` is *past* the last row (you cannot retrieve columns there). Returns `true`, if the requested destination is reachable.

- **override fun moveToFirst(): Boolean**

  Moves the cursor to the first row. Returns `true` if the first row exists.

- **override fun moveToLast(): Boolean**

  Moves the cursor to the last row. Returns `false` if the cursor is empty.

- **override fun moveToPrevious(): Boolean**

  Move the cursor to the previous row. Returns `false` if the cursor is already before the first row.

- **override fun moveToNext(): Boolean**

  Move the cursor to the next row. Returns `false` if the cursor is already past the last entry.

- **override fun move(offset: Int): Boolean**

  Move the cursor by a relative amount, forward (positive argument) or backward (negative argument), from the current position. This will be clamped to `-1` or `count`. This returns `true` if the requested destination was reachable (implies `false` if clamped).

- **override fun getPosition(): Int**

  Returns the current position (zero based) of the cursor in the row set. It ranges from `-1` to `count`, with the limits pointing to nonexisting rows.

- **override fun isFirst(): Boolean**

  Returns whether the cursor is pointing to the first row.

- **override fun isLast(): Boolean**

  Returns whether the cursor is pointing to the last row.

- **override fun isBeforeFirst(): Boolean**

  Returns whether the cursor is pointing to the position before the first row (you cannot retrieve columns there).

- **override fun isAfterLast(): Boolean**

  Returns whether the cursor is pointing to the position after the last row (you cannot retrieve columns there).

Data Retrieval

- **override fun getColumnCount(): Int**

  Returns the number of columns.

- **override fun getColumnNames(): Array<String>**

  Returns the ordered `string` array with the names of all the columns in the result set.

- **override fun getType(columnIndex: Int): Int**

  The data type of the specified column (zero based). One of: `FIELD_TYPE_NULL`, `FIELD_TYPE_INTEGER`, `FIELD_TYPE_FLOAT`, `FIELD_TYPE_STRING`, or `FIELD_TYPE_BLOB`.

- **override fun getColumnName(columnIndex: Int): String**

  Returns the column name at the given zero-based column index.

- **override fun getColumnIndex(columnName: String?): Int**

  Returns the zero-based index for the given column name, or -1 if the column doesn't exist.

- **override fun getColumnIndexOrThrow(columnName: String?): Int**

  Returns the zero-based index for the given column name, or throws an `IllegalArgumentException` if the column doesn't exist.

- **override fun isNull(columnIndex: Int): Boolean**

  The value of the requested column (zero based) is `null`.

- **override fun getInt(columnIndex: Int): Int**

  The value of the requested column (zero based) as an `int`.

- **override fun getShort(columnIndex: Int): Short**

  The value of the requested column (zero based) as a short `int`.

- **override fun getLong(columnIndex: Int): Long**

  The value of the requested column (zero based) as a long.

- **override fun getFloat(columnIndex: Int): Float**

  The value of the requested column (zero based) as a float.

- **override fun getDouble(columnIndex: Int): Double**

  The value of the requested column (zero based) as a double.

- **override fun getString(columnIndex: Int): String**

  The value of the requested column (zero based) as a string.

- **override fun copyStringToBuffer(columnIndex: Int, buffer: CharArrayBuffer?)**

  Retrieves the requested column text (index zero based) and stores it in the buffer provided. Other than suggested by the argument signature, the buffer must not be null.

- **override fun getBlob(columnIndex: Int): ByteArray**

  Returns the value of the requested column (zero based) as a byte array.

If for retrieving columns the types don't match, and whether this method throws an exception when the column value is null, is implementation-defined. Usually you will add type converters where applicable.

Listeners

- **override fun setNotificationUri(cr: ContentResolver?, uri: Uri?**

  Clients use this to register to watch a content URI for changes. This can be the URI of a specific data row (for example, "`content://my_provider_type/23`") or of a generic URI for a content type. The receiver of the changes is mediated by the `ContentResolver` parameter. The listener attached to this resolver will be notified.

- **override fun getNotificationUri(): Uri**

  Return the URI at which notifications of changes in this cursor's data will be delivered. Returns a URI that can be used with `ContentResolver.registerContentObserver()` to find out about changes to this cursor's data, or `null` if no notification URI has been set.

- **override fun registerContentObserver(observer: ContentObserver?**

  Register an observer that is called when changes happen to the content backing this cursor.

- **override fun unregisterContentObserver(observer: ContentObserver?**

  Unregister an observer that has previously been registered with this cursor.

- **override fun registerDataSetObserver(observer: DataSetObserver?**

  Register an observer that is called when changes happen to the contents of this cursor's data set.

- **override fun unregisterDataSetObserver(observer: DataSetObserver?**

  Unregister an observer that has previously been registered.

Various

- **override fun getWantsAllOnMoveCalls(): Boolean**

  This sneaked into the interface from its base implementation `AbstractCursor`. The method `AbstractCursor.onMove()` will be called across processes only if this method returns `true`.

Features

The following are values for the `android:name` attribute of the `<uses-feature>` element inside the manifest. Adding such elements to `AndroidManifest.xml` expresses a requirement.

Hardware Features

Here we describe the hardware features supported by API level 26 (Android 8). For prior versions or the history of features, please consult the online documentation.

Audio Hardware Features

- **android.hardware.audio.low_latency**

  Use the low-latency audio pipeline for sound input or output, reducing lags and delays.

- **android.hardware.audio.output**

  The device must support some capability for output sound.

- **android.hardware.audio.pro**

  Expresses the requirement for the device to support high-end audio functionality.

- **android.hardware.microphone**

  The device must support some audio capture mechanism like a microphone.

Bluetooth Hardware Features

- **android.hardware.bluetooth**

  The device must support Bluetooth.

- **android.hardware.bluetooth_le**

  The device must support low-energy Bluetooth.

Camera Hardware Features

- **android.hardware.camera**

  The device must have a back-facing camera.

- **android.hardware.camera.any**

  The device must have any camera, including external cameras that can be connected to it.

- **android.hardware.camera.autofocus**

  The camera must have an autofocus. This implies `android.hardware.camera` unless that one is declared with `android:required="false"`.

- **android.hardware.camera.capability.manual_post_processing**

  The camera must have a `MANUAL_POST_PROCESSING` feature. Allow for the app to override the camera's automatic white balance functionality. For that aim, in class `CaptureRequest`, use `android.colorCorrection.transform`, `android.colorCorrection.gains`, and `android.colorCorrection.mode` of `TRANSFORM_MATRIX`.

- **android.hardware.camera.capability.manual_sensor**

  The camera must support the `MANUAL_SENSOR` feature, which implies auto-exposure locking. It must be possible to manually set a fixed exposure time and a fixed sensitivity.

- **android.hardware.camera.capability.raw**

  The camera must be able to save raw image files, including the necessary DNG metadata.

- **android.hardware.camera.external**

  The device provides a capability to communicate with externally connected cameras.

- **android.hardware.camera.flash**

  The camera must have a flash. This implies `android.hardware.camera` unless that one is declared with `android:required="false"`.

- **android.hardware.camera.front**

  The device must have a front-facing camera. This implies `android.hardware.camera` unless that one is declared with `android:required="false"`.

- **android.hardware.camera.level.full**

  One of the cameras must have `FULL`-level image-capturing support. This means burst-capture capabilities, per-frame control, and manual post-processing control.

Device UI Hardware Features

- **android.hardware.type.automotive**

  The app expects to be shown inside a vehicle. This implies hard buttons, touch, rotary controllers, and mouse-like interfaces.

- **android.hardware.type.television**

  Deprecated. Use `android.software.leanback` instead. The device is a television-like apparatus, the screen is big, and the user sits far away from it. That means a mouse or touch capabilities don't get used.

- **android.hardware.type.watch**

  The device is a wearable like a watch.

Fingerprint Hardware Features

- **android.hardware.fingerprint**

  The device has a fingerprint scanner.

Gamepad Hardware Features

- **android.hardware.gamepad**

  The device has a game controller or a gamepad input.

Infrared Hardware Features

- **android.hardware.consumerir**

  The device has an infrared sensor.

Location Hardware Features

- **android.hardware.location**

  The device provides for features for determining location. One of or a combination of GPS location, network location, or cell location.

- **android.hardware.location.gps**

  The device provides for features for determining precise location via GPS. This implies `android.hardware.location`, unless the latter got marked with `android:required="false"`.

- **android.hardware.location.network**

  The device provides for features for determining the coarse location via a network. It implies `android.hardware.location`, unless the latter got marked with `android:required="false"`.

NFC Hardware Features

- **android.hardware.nfc**

  Deprecated. The device has NFC capabilities.

OpenGL ES Hardware Features

- **android.hardware.opengles.aep**

  The device supports the OpenGL ES Android Extension Pack.

Sensor Hardware Features

- **android.hardware.sensor.accelerometer**

  The device has an accelerometer.

- **android.hardware.sensor.ambient_temperature**

  The device can detect the environmental temperature.

- **android.hardware.sensor.barometer**

  The device has a barometer.

- **android.hardware.sensor.compass**

  The device has a compass.

- **android.hardware.sensor.gyroscope**

  The device has a gyroscope to detect rotation and twist.

- **android.hardware.sensor.hifi_sensors**

  The device has high-fidelity (Hi-Fi) sensors.

- **android.hardware.sensor.heartrate**

  The device has a heart rate monitor.

- **android.hardware.sensor.heartrate.ecg**

  The device has a elcardiogram (ECG) heart rate sensor.

- **android.hardware.sensor.light**

  The device has a light sensor.

- **android.hardware.sensor.proximity**

  The device has a proximity sensor.

- **android.hardware.sensor.relative_humidity**

  The device has a relative humidity sensor.

- **android.hardware.sensor.stepcounter**

  The device has a step counter.

- **android.hardware.sensor.stepdetector**

  The device has a step detector.

Screen Hardware Features

- **android.hardware.screen.landscape**

- **android.hardware.screen.portrait**

  Specifies that the device must be able to show the app in either landscape or portrait mode, or both. If your app supports both orientations, then you don't need to declare either feature.

Telephony Hardware Features

- **android.hardware.telephony**

  The device can act as a phone.

- **android.hardware.telephony.cdma**

  The device provides a Code Division Multiple Access (CDMA) telephony radio system. This implies `android.hardware.telephony`, unless the latter is marked with `android:required="false"`.

- **android.hardware.telephony.gsm**

  The device provides a Global System for Mobile Communications (GSM) telephony radio system. This implies `android.hardware.telephony`, unless the latter is marked with `android:required="false"`.

Touchscreen Hardware Features

- **android.hardware.faketouch**

  The device provides either real or fake touch features. Apps require that by default. The app uses basic touch interaction events, such as tapping and dragging, or their simulated counterpart.

- **android.hardware.faketouch.multitouch.distinct**

  The device provides for the capability to distinguish between two or more fingers. A device that provides a two-finger touch trackpad for cursor movement can support this feature.

- **android.hardware.faketouch.multitouch.jazzhand**

  The app tracks five or more distinct "fingers" on a fake touch interface. A device that provides a five-finger touch trackpad for cursor movement can support this feature.

- **android.hardware.touchscreen**

  The device must support more than basic touch events. If your app instead wants to allow for simulated touch events via `faketouch`, you must explicitly declare this element and add `android:required="false"`; otherwise, this feature is required by default.

- **android.hardware.touchscreen.multitouch**

  This is a superset of the `android.hardware.touchscreen` feature. The device must support basic two-point multitouch capabilities, but it needs not track touches independently.

- **android.hardware.touchscreen.multitouch.distinct**

  This is a superset of `android.hardware.touchscreen.multitouch`. The device must allow apps to track multitouch gesture points independently.

- **android.hardware.touchscreen.multitouch.jazzhand**

  The device must allow for using five or more multitouch points independently.

USB Hardware Features

- **android.hardware.usb.accessory**

  The device must be able to connect to USB hosts.

- **android.hardware.usb.host**

  The device must be able to serve as a USB host.

Wi-Fi Hardware Features

- **android.hardware.wifi**

  The device must use 802.11 networking (Wi-Fi) features.

- **android.hardware.wifi.direct**

  The device must allow for Wi-Fi Direct networking.

Software Features

The following paragraphs represent software features your app may declare to be required.

Communication software features

- **android.software.sip**

  The device must provide Session Initiation Protocol (SIP) services.

- **android.software.sip.voip**

  The device must support SIP-based Voice over Internet Protocol (VoIP) services.

- **android.software.webview**

  The app wants to display content from the Internet.

Custom Input Software Features

- **android.software.input_methods**

  The app wants to define a new input method, as declared in `InputMethodService`.

Device Management Software Features

- **android.software.backup**

  The app wants to use logic to handle backup and restore operations.

- **android.software.device_admin**

  The app wants to enforce a device policy in an administration workflow.

- **android.software.managed_users**

  The app supports secondary users and managed profiles.

- **android.software.securely_removes_users**

  The app wants to be able to permanently remove users and their associated data.

- **android.software.verified_boot**

  The app wants to handle results from the device's verified boot feature. This detects whether the device's configuration changes during a restart operation.

Media Software Features

- **android.software.midi**

  The app wants to address a MIDI interface to talk to connected instruments.

- **android.software.print**

  The app wants to use a printer.

- **android.software.leanback**

  The app needs a large screen, such as a TV device.

- **android.software.live_tv**

  The app wants to stream live television programs.

Screen Interface Software Features

- **android.software.app_widgets**

  The app uses app widgets.

- **android.software.home_screen**

  The app provides a replacement for the device's home screen.

- **android.software.live_wallpaper**

  The app uses or provides animated wallpapers.

Permissions

Table A-15 lists permissions.

Table A-15. *Intent Actions for Broadcasts*

Name	Description
`ACCESS_CHECKIN_PROPERTIES`	Allows read-write access to the "properties" table in the check-in database to change values that get uploaded.
`ACCESS_COARSE_LOCATION`	Allows an app to access an approximate location.
`ACCESS_FINE_LOCATION`	Allows an app to access a precise location.
`trs ACCESS_LOCATION_EXTRA_COMMANDS`	Allows an application to access extra location provider commands.
`ACCESS_NETWORK_STATE`	Allows applications to access information about networks.
`ACCESS_NOTIFICATION_POLICY`	Marker permission for applications that want to access the notification policy.
`ACCESS_WIFI_STATE`	Allows applications to access information about Wi-Fi networks.

(*continued*)

Table A-15. (*continued*)

Name	Description
ACCOUNT_MANAGER	Allows applications to call into AccountAuthenticators.
ADD_VOICEMAIL	Allows an application to add voicemails into the system.
ANSWER_PHONE_CALLS	Allows the app to answer an incoming phone call.
BATTERY_STATS	Allows an application to collect battery statistics.
BIND_ACCESSIBILITY_SERVICE	Must be required by an AccessibilityService to ensure that only the system can bind to it.
BIND_APPWIDGET	Allows an application to tell the AppWidget service which application can access AppWidget's data.
BIND_AUTOFILL_SERVICE	Must be required by a AutofillService to ensure that only the system can bind to it.
BIND_CARRIER_MESSAGING_SERVICE	This constant was deprecated in API level 23. Use BIND_CARRIER_SERVICES instead.
BIND_CARRIER_SERVICES	The system process that is allowed to bind to services in carrier apps will have this permission.
BIND_CHOOSER_TARGET_SERVICE	Must be required by a ChooserTargetService to ensure that only the system can bind to it.
BIND_CONDITION_PROVIDER_SERVICE	Must be required by a ConditionProviderService to ensure that only the system can bind to it.
BIND_DEVICE_ADMIN	Must be required by device administration receiver to ensure that only the system can interact with it.
BIND_DREAM_SERVICE	Must be required by DreamService to ensure that only the system can bind to it.
BIND_INCALL_SERVICE	Must be required by InCallService to ensure that only the system can bind to it.
BIND_INPUT_METHOD	Must be required by InputMethodService to ensure that only the system can bind to it.
BIND_MIDI_DEVICE_SERVICE	Must be required by MidiDeviceService to ensure that only the system can bind to it.
BIND_NFC_SERVICE	Must be required by HostApduService or OffHostApduService to ensure that only the system can bind to it.
BIND_NOTIFICATION_LISTENER_SERVICE	Must be required by NotificationListenerService to ensure that only the system can bind to it.
BIND_PRINT_SERVICE	Must be required by PrintService to ensure that only the system can bind to it.
BIND_QUICK_SETTINGS_TILE	Allows an application to bind to third-party quick settings tiles.
BIND_REMOTEVIEWS	Must be required by RemoteViewsService to ensure that only the system can bind to it.

(*continued*)

Table A-15. (*continued*)

Name	Description
BIND_SCREENING_SERVICE	Must be required by a CallScreeningService to ensure that only the system can bind to it.
BIND_TELECOM_CONNECTION_SERVICE	Must be required by a ConnectionService to ensure that only the system can bind to it.
BIND_TEXT_SERVICE	Must be required by a TextService.
BIND_TV_INPUT	Must be required by a TvInputService to ensure that only the system can bind to it.
BIND_VISUAL_VOICEMAIL_SERVICE	Must be required by a link to VisualVoicemailService to ensure that only the system can bind to it.
BIND_VOICE_INTERACTION	Must be required by VoiceInteractionService to ensure that only the system can bind to it.
BIND_VPN_SERVICE	Must be required by VpnService to ensure that only the system can bind to it.
BIND_VR_LISTENER_SERVICE	Must be required by VrListenerService to ensure that only the system can bind to it.
BIND_WALLPAPER	Must be required by WallpaperService to ensure that only the system can bind to it.
BLUETOOTH	Allows applications to connect to paired Bluetooth devices.
BLUETOOTH_ADMIN	Allows applications to discover and pair Bluetooth devices.
BLUETOOTH_PRIVILEGED	Allows applications to pair Bluetooth devices without user interaction and to allow or disallow phonebook access or message access.
BODY_SENSORS	Allows an application to access data from sensors that the user uses to measure what is happening inside their body, such as heart rate.
BROADCAST_PACKAGE_REMOVED	Allows an application to broadcast a notification that an application package has been removed.
BROADCAST_SMS	Allows an application to broadcast an SMS receipt notification.
BROADCAST_STICKY	Allows an application to broadcast sticky intents.
BROADCAST_WAP_PUSH	Allows an application to broadcast a WAP PUSH receipt notification.
CALL_PHONE	Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call.
CALL_PRIVILEGED	Allows an application to call any phone number, including emergency numbers, without going through the Dialer user interface for the user to confirm the call being placed.

(*continued*)

Table A-15. (*continued*)

Name	Description
CAMERA	Required to be able to access the camera device.
CAPTURE_AUDIO_OUTPUT	Allows an application to capture audio output.
CAPTURE_SECURE_VIDEO_OUTPUT	Allows an application to capture secure video output.
CAPTURE_VIDEO_OUTPUT	Allows an application to capture video output.
CHANGE_COMPONENT_ENABLED_STATE	Allows an application to change whether an application component (other than its own) is enabled.
CHANGE_CONFIGURATION	Allows an application to modify the current configuration, such as locale.
CHANGE_NETWORK_STATE	Allows applications to change network connectivity state.
CHANGE_WIFI_MULTICAST_STATE	Allows applications to enter Wi-Fi multicast mode.
CHANGE_WIFI_STATE	Allows applications to change Wi-Fi connectivity state.
CLEAR_APP_CACHE	Allows an application to clear the caches of all installed applications on the device.
CONTROL_LOCATION_UPDATES	Allows enabling/disabling location update notifications from the radio.
DELETE_CACHE_FILES	Allows an application to delete cache files.
DELETE_PACKAGES	Allows an application to delete packages.
DIAGNOSTIC	Allows applications to RW to diagnostic resources.
DISABLE_KEYGUARD	Allows applications to disable the keyguard if it is not secure.
DUMP	Allows an application to retrieve state dump information from system services.
EXPAND_STATUS_BAR	Allows an application to expand or collapse the status bar.
FACTORY_TEST	Runs as a manufacturer test application, running as the root user.
GET_ACCOUNTS	Allows access to the list of accounts in the Accounts Service.
GET_ACCOUNTS_PRIVILEGED	Allows access to the list of accounts in the Accounts Service.
GET_PACKAGE_SIZE	Allows an application to find out the space used by any package.
GET_TASKS	This constant was deprecated in API level 21. No longer enforced.
GLOBAL_SEARCH	This permission can be used on content providers to allow the global search system to access their data.

(*continued*)

Table A-15. (*continued*)

Name	Description
INSTALL_LOCATION_PROVIDER	Allows an application to install a location provider into the Location Manager.
INSTALL_PACKAGES	Allows an application to install packages.
INSTALL_SHORTCUT	Allows an application to install a shortcut in Launcher.
INSTANT_APP_FOREGROUND_SERVICE	Allows an instant app to create foreground services.
INTERNET	Allows applications to open network sockets.
KILL_BACKGROUND_PROCESSES	Allows an application to call killBackgroundProcesses(String).
LOCATION_HARDWARE	Allows an application to use location features in hardware, such as the Geofencing API.
MANAGE_DOCUMENTS	Allows an application to manage access to documents, usually as part of a document picker.
MANAGE_OWN_CALLS	Allows a calling application that manages its own calls through the self-managed ConnectionService APIs.
MASTER_CLEAR	Not for use by third-party applications.
MEDIA_CONTENT_CONTROL	Allows an application to know what content is playing and control its playback.
MODIFY_AUDIO_SETTINGS	Allows an application to modify global audio settings.
MODIFY_PHONE_STATE	Allows modification of the telephony state: power on, mmi, etc.
MOUNT_FORMAT_FILESYSTEMS	Allows formatting file systems for removable storage.
MOUNT_UNMOUNT_FILESYSTEMS	Allows mounting and unmounting file systems for removable storage.
NFC	Allows applications to perform I/O operations over NFC.
PACKAGE_USAGE_STATS	Allows an application to collect component usage statistics. Declaring the permission implies intention to use the API and the user of the device can grant permission through the Settings application.
PERSISTENT_ACTIVITY	This constant was deprecated in API level 9. This functionality will be removed in the future; please do not use it. It allows an application to make its activities persistent.
PROCESS_OUTGOING_CALLS	Allows an application to see the number being dialed during an outgoing call with the option to redirect the call to a different number or abort the call altogether.
READ_CALENDAR	Allows an application to read the user's calendar data.
READ_CALL_LOG	Allows an application to read the user's call log.

(*continued*)

Table A-15. (*continued*)

Name	Description
READ_CONTACTS	Allows an application to read the user's contacts data.
READ_EXTERNAL_STORAGE	Allows an application to read from external storage.
READ_FRAME_BUFFER	Allows an application to take screenshots and more generally get access to the frame buffer data.
READ_INPUT_STATE	This constant was deprecated in API level 16. The API that used this permission has been removed.
READ_LOGS	Allows an application to read the low-level system log files.
READ_PHONE_NUMBERS	Allows read access to the device's phone number(s).
READ_PHONE_STATE	Allows read-only access to the phone state, including the phone number of the device, current cellular network information, status of any ongoing calls, and list of any PhoneAccounts registered on the device.
READ_SMS	Allows an application to read SMS messages.
READ_SYNC_SETTINGS	Allows applications to read the sync settings.
READ_SYNC_STATS	Allows applications to read the sync stats.
READ_VOICEMAIL	Allows an application to read voicemails in the system.
REBOOT	Required to be able to reboot the device.
RECEIVE_BOOT_COMPLETED	Allows an application to receive the ACTION_BOOT_COMPLETED that is broadcast after the system finishes booting.
RECEIVE_MMS	Allows an application to monitor incoming MMS messages.
RECEIVE_SMS	Allows an application to receive SMS messages.
RECEIVE_WAP_PUSH	Allows an application to receive WAP push messages.
RECORD_AUDIO	Allows an application to record audio.
REORDER_TASKS	Allows an application to change the Z-order of tasks.
REQUEST_COMPANION_RUN_IN_BACKGROUND	Allows a companion app to run in the background.
REQUEST_COMPANION_USE_DATA_IN_BACKGROUND	Allows a companion app to use data in the background.
REQUEST_DELETE_PACKAGES	Allows an application to request deleting packages.
REQUEST_IGNORE_BATTERY_OPTIMIZATIONS	Permission an application must hold in order to use ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS.
REQUEST_INSTALL_PACKAGES	Allows an application to request installing packages.
RESTART_PACKAGES	This constant was deprecated in API level 8. The restartPackage(String) API is no longer supported.
SEND_RESPOND_VIA_MESSAGE	Allows an application (Phone) to send a request to other applications to handle the respond-via-message action during incoming calls.

(*continued*)

Table A-15. (*continued*)

Name	Description
SEND_SMS	Allows an application to send SMS messages.
SET_ALARM	Allows an application to broadcast an intent to set an alarm for the user.
SET_ALWAYS_FINISH	Allows an application to control whether activities are immediately finished when put in the background.
SET_ANIMATION_SCALE	Modify the global animation scaling factor.
SET_DEBUG_APP	Configure an application for debugging.
SET_PREFERRED_APPLICATIONS	This constant was deprecated in API level 7. This is no longer useful; see `addPackageToPreferred(String)` for details.
SET_PROCESS_LIMIT	Allows an application to set the maximum number of (not needed) application processes that can be running.
SET_TIME	Allows applications to set the system time.
SET_TIME_ZONE	Allows applications to set the system time zone.
SET_WALLPAPER	Allows applications to set the wallpaper.
SET_WALLPAPER_HINTS	Allows applications to set the wallpaper hints.
SIGNAL_PERSISTENT_PROCESSES	Allows an application to request that a signal be sent to all persistent processes.
STATUS_BAR	Allows an application to open, close, or disable the status bar and its icons.
SYSTEM_ALERT_WINDOW	Allows an app to create windows using the type `TYPE_APPLICATION_OVERLAY`, shown on top of all other apps.
TRANSMIT_IR	Allows using the device's IR transmitter, if available.
UNINSTALL_SHORTCUT	This permission is no longer supported.
UPDATE_DEVICE_STATS	Allows an application to update device statistics.
USE_FINGERPRINT	Allows an app to use fingerprint hardware.
USE_SIP	Allows an application to use SIP service.
VIBRATE	Allows access to the vibrator.
WAKE_LOCK	Allows using PowerManager `WakeLocks` to keep processor from sleeping or screen from dimming.
WRITE_APN_SETTINGS	Allows applications to write the APN settings.
WRITE_CALENDAR	Allows an application to write the user's calendar data.
WRITE_CALL_LOG	Allows an application to write (but not read) the user's call log data.
WRITE_CONTACTS	Allows an application to write the user's contacts data.
WRITE_EXTERNAL_STORAGE	Allows an application to write to external storage.

(*continued*)

Table A-15. (*continued*)

Name	Description
WRITE_GSERVICES	Allows an application to modify the Google service map.
WRITE_SECURE_SETTINGS	Allows an application to read or write the secure system settings.
WRITE_SETTINGS	Allows an application to read or write the system settings.
WRITE_SYNC_SETTINGS	Allows applications to write the sync settings.
WRITE_VOICEMAIL	Allows an application to modify and remove existing voicemails in the system.

The System Intent Filters

The following are the system intent filters. You can use them to launch activities and services of system apps; see Chapter 3.

Unless otherwise noted, for all actions prepend `android.intent.action.`, and for all categories prepend `android.intent.category.`.

> **Note** These filters have been extracted by a script with the help of Apktool, which you can find at GitHub.

Table A-16. Browser2 Activity Intent Filters

Action	Category	Data	Act
MAIN	LAUNCHER		1
VIEW	DEFAULT	scheme:http	(1)
	BROWSABLE	scheme:https	
VIEW	DEFAULT	scheme:http	(1)
	BROWSABLE	scheme:https	
		mimeType:text/html	
		mimeType:text/plain	
		mimeType:application/ xhtml+xml	
		mimeType:application/ vnd.wap.xhtml+xml	

(1) *WebViewBrowserActivity*

Table A-17. CalendarGooglePrebuilt Activity Intent Filters

Action	Category	Data	Act
MAIN	LAUNCHER		(1)
EDIT INSERT VIEW	DEFAULT	mimeType: vnd.android.cursor.item/event	(1)
EDIT INSERT	DEFAULT	mimeType: vnd.android.cursor.dir/event	(1)
VIEW	DEFAULTBROWSABLE	scheme:http scheme:https host:www.google.com pathPrefix:/calendar/event pathPattern: /calendar/hosted/.*/event	(1)
VIEW	DEFAULTBROWSABLE	scheme:http scheme:https host:calendar.google.com pathPrefix:/calendar/event pathPattern: /calendar/hosted/.*/event	(1)
EDIT VIEW	DEFAULT	host:calendar scheme:settings	(2)
SEARCH			(3)
VIEW	DEFAULT	scheme:http scheme:https scheme:content scheme:file scheme: host:* pathPattern:.*\.ics	(4)
VIEW	DEFAULT	scheme:http scheme:https scheme:content scheme:file scheme: host:* mimeType:text/calendar	(4)

(*continued*)

Table A-17. (*continued*)

Action	Category	Data	Act
VIEW	DEFAULT	scheme:http scheme:https scheme:content scheme:file scheme: host:* mimeType:application/ics	(4)
VIEW	DEFAULT	scheme:http scheme:https scheme:content scheme:file scheme: host:* mimeType: application/octet-stream pathPattern:.*\.ics	
MANAGE_NETWORK_USAGE	DEFAULT		(5)
MAIN	DEFAULT NOTIFICATION_PREFERENCES		(5)
MAIN	DEFAULT LAUNCHER APP_CALENDAR		(6)
VIEW	DEFAULT	mimeType:time/epoch host:com.android.calendar scheme:content	(6)
(7).EVENT_EDIT (7).EVENT_INSERT (7).EVENT_VIEW			(6)
(7).FIND_TIME	DEFAULT		(6)

(1) *LaunchInfoActivity*

(2) *GoogleCalendarSettingsActivity*

(3) *SearchActivity*

(4) *ICalLauncher*

(5) *CalendarPublicPreferenceActvity*

(6) *AllInOneActivity*

(7) *com.google.android.calendar*

Table A-18. Camera2 Activity Intent Filters

Action	Category	Data	Act
(1).STILL_IMAGE_CAMERA	DEFAULT		(2)
MAIN	DEFAULT		(2)
(1).IMAGE_CAPTURE]	DEFAULT		(3)
(1).STILL_IMAGE_CAMERA_SECURE	DEFAULT		(4)
(1).IMAGE_CAPTURE_SECURE	DEFAULT		(4)
MAIN	DEFAULT LAUNCHER		(5)
(1).VIDEO_CAMERA	DEFAULT		(6)
(1).VIDEO_CAPTURE	DEFAULT		(6)

(1) *android.media.action*

(2) *CameraActivity*

(3) *CaptureActivity*

(4) *SecureCameraActivity*

(5) *CameraLauncher*

(6) *VideoCamera*

Table A-19. CaptivePortalLogin Activity Intent Filters

Action	Category	Data	Act
(1).CAPTIVE_PORTAL	DEFAULT		(2)

(1) android.net.*conn*

(2) *CaptivePortalLoginActivity*

Table A-20. CertInstaller Activity Intent Filters

Action	Category	Data	Act
(1).INSTALL	DEFAULT		(2)
VIEW	DEFAULT	mimeType:	(2)
		application/x-x509-ca-cert	
		mimeType:	
		application/x-x509-user-cert	
		mimeType:	
		application/x-x509-server-cert	
		mimeType:	
		application/x-pkcs12	
		mimeType:	
		application/application/x-pem-file	
		mimeType:	
		application/pkix-cert	
		mimeType:	
		application/x-wifi-config	
(1).INSTALL_AS_USER	DEFAULT		(3)

(1) *android.credentials*

(2) *CertInstallerMain*

(3) *InstallCertAsUser*

Table A-21. Chrome Activity Intent Filters

Action	Category	Data	Act
(8)	(9).DAYDREAM (9).CARDBOARD		(1)
(10).ADDBOOKMARK	DEFAULT		(2)
(11).ACTION_START_ WEBAPP	DEFAULT		(3)
SEND	DEFAULT	mimeType:text/plain	(4)
SEND	DEFAULT	mimeType:text/plain	(5)
(12).WEBVIEW_LICENSE	DEFAULT		(6)
MAIN	DEFAULT		(7)
	LAUNCHER		
	BROWSABLE		
	APP_BROWSER		
	NOTIFICATION_PREFERENCES		

(*continued*)

Table A-21. (*continued*)

Action	Category	Data	Act
VIEW	DEFAULT BROWSABLE	scheme:googlechrome scheme:http scheme:https scheme:about scheme:javascript	(7)
VIEW	DEFAULT BROWSABLE	scheme:googlechrome scheme:http scheme:https scheme:about scheme:content scheme:javascript mimeType:text/html mimeType:text/plain mimeType: application/xhtml+xml	(7)
VIEW	DEFAULT	scheme:file scheme:content mimeType: multipart/related	(7)
VIEW	DEFAULT BROWSABLE	scheme:file scheme:content host:* pathPattern:/.*\.mhtml pathPattern:/.*\.mht	(7)
VIEW	DEFAULT BROWSABLE	scheme:file scheme:content host:* mimeType:*/* pathPattern:/.*\.mhtml pathPattern:/.*\.mht	(7)
MEDIA_SEARCH	DEFAULT		(7)

(*continued*)

Table A-21. (*continued*)

Action	Category	Data	Act
(13).VOICE_SEARCH_RESULTS	DEFAULT		(7)
(14).NDEF_DISCOVERED	DEFAULT	scheme:http scheme:https	(7)
SEARCH			(7)
(15).HOVER			(7)

(1) *ChromeTabbedActivity*

(2) *BookmarkAddActivity*

(3) *WebappLauncherActivity*

(4) *PrintShareActivity*

(5) *PhysicalWebShareActivity*

(6) LicenseActivity

(7) *Main*

(8) *org.chromium.chrome.browser.dummy.action*

(9) *com.google.intent.category*

(10) *com.android.chrome*

(11) *com.google.android.apps.chrome.webapps.WebappManager*

(12) *android.settings*

(13) *android.speech.action*

(14) *android.nfc.action*

(15) *com.sec.android.airview*

Table A-22. CompanionDeviceManager Activity Intent Filters

Action	Category	Data	Act
(1).START_DISCOVERY	DEFAULT		(2)

(1) *android.companiondevice*

(2) *DeviceChooserActivity*

Table A-23. CtsShimPrebuilt Activity Intent Filters

Action	Category	Data	Act
SEARCH	INFO		(1)
VIEW	BROWSABLE		(1)
SEND			(1)
SEND_MULTIPLE			(1)
SENDTO			(1)

(1) *InstallPriority*

Table A-24. CustomLocale (Only Virtual Device!) Activity Intent Filters

Action	Category	Data	Act
MAIN	LAUNCHER		(1)

(1) *CustomLocaleActivity*

Table A-25. Development (Only Virtual Device!) Activity Intent Filters

Action	Category	Data	Act
MAIN	LAUNCHER		Development
MAIN	TEST		PackageBrowser
MAIN	TEST		PointerLocation
MAIN	TEST		AccountsTester
MAIN	TEST		SyncAdapterDriver
MAIN	TEST		Connectivity
MAIN	TEST		InstrumentationList
MAIN	TEST		MediaScannerActivity
MAIN	TEST		RunningProcesses
VIEW	DEFAULT		ProcessInfo
VIEW	DEFAULT		AppHwPref
(1).VIEW_PERMISSION	DEFAULT		PermissionDetails
MAIN	TEST		BadBehaviorActivity
MAIN	TEST		CacheAbuser
MAIN	TEST		ConfigurationViewer

(1) *com.android.development*

Table A-26. DownloadProviderUi Activity Intent Filters

Action	Category	Data	Act
(1).MANAGE_DOCUMENT	DEFAULT	scheme:content mimeType:*/* host:com.android.providers. downloads.documents	(2)

(1) *android.provider.action*

(2) *TrampolineActivity*

Table A-27. Drive Activity Intent Filters

Action	Category	Data	Act
MAIN (30).SEARCH_ACTION	LAUNCHER DEFAULT		(1)
(31).SEARCH_SHORTCUT_ACTION			
VIEW			(2)
SEARCH			(2)
SEND SEND_MULTIPLE	DEFAULT	mimeType:*/*	(3)
SEND	DEFAULT	scheme:file	(3)
(32).WELCOME	DEFAULT		(4)
VIEW			(5)
BUG_REPORT			(6)
VIEW	DEFAULT		(7)
MAIN	PREFERENCE		(8)
(32).DRIVE_STORAGE	DEFAULT		(9)
VIEW			(10)
PICK	DEFAULT		(11)
SEND	DEFAULT	mimeType:text/plain	(12)
MAIN	NOTIFICATION_ PREFERENCES		(13)
(33).NOTIFICATION_SETTINGS	DEFAULT		(14)
(33).NOTIFICATION_HOME	DEFAULT		(15)
CREATE_SHORTCUT	DEFAULT		(16)
CREATE_SHORTCUT	DEFAULT		(17)
(34).APPWIDGET_CONFIGURE			(18)
QUICK_VIEW	DEFAULT	scheme:projector-id	(19)
QUICK_VIEW	DEFAULT		(19)

(*continued*)

Table A-27. (*continued*)

Action	Category	Data	Act
QUICK_VIEW	DEFAULT	scheme:file	(19)
		scheme:content	
		scheme:http	
QUICK_VIEW	DEFAULT	scheme:file	(19)
		scheme:content	
		scheme:http	
		mimeType:*/*	
VIEW	DEFAULT	scheme:file	(20)
	BROWSABLE	scheme:content	
		scheme:http	
		mimeType:application/pdf	
VIEW	DEFAULT	scheme:http	(20)
	BROWSABLE	pathPattern:.*\\.pdf host:*	
VIEW	DEFAULT	scheme:file pathPattern:.*\\.pdf	(20)
VIEW	DEFAULT	(43)	(21)
VIEW	DEFAULT	scheme:content	(21)
		mimeType:(37).document	
		host:(35)	
VIEW	DEFAULT	scheme:content	(21)
		mimeType:(37).presentation	
		host:(35)	
VIEW	DEFAULT	scheme:content	(21)
		mimeType:(37).spreadsheet	
		host:(35)	
VIEW			(22)
SEARCH			(22)
GET_CONTENT	DEFAULT	mimeType:*/*	(23)
	OPENABLE		
PICK	DEFAULT		(24)
VIEW			(25)

(*continued*)

Table A-27. (*continued*)

Action	Category	Data	Act
VIEW	DEFAULT BROWSABLE	(40)	(26)
VIEW	DEFAULT BROWSABLE	(41)	(27)
VIEW	DEFAULT BROWSABLE	(42)	(27)
VIEW	DEFAULT BROWSABLE	(39)	(28)
VIEW (32).REQUEST_ACCESS	DEFAULT BROWSABLE	s:http + h:drive.google.com s:https + h:drive.google.com s:http + h:icing.drive.google.com s:https + h:icing.drive.google.com	(29)
VIEW (32).REQUEST_ACCESS	DEFAULT BROWSABLE	(38)	(29)

Inside *data*: *s* = *scheme*, *m* = *mimeType*, *h* = *host*, *pp* = *pathPattern*

(1) *NewMainProxyActivity*

(2) *DocumentOpenerActivity*

(3) *UploadMenuActivity*

(4) *WelcomeActivity*

(5) *DocumentOpenerActivityDelegate*

(6) *ErrorNotificationActivity*

(7) *LegacyPrintActivity*

(8) *DocsPreferencesActivity*

(9) *PaymentsActivity*

(10) *TestFragmentActivity*

(11) *PickEntryActivity*

(12) *SendTextToClipboardActivity*

(13) *NotificationPreferencesActivity*

(14) *ExportedNotificationPreferencesActivity*

(15) *ExportedNotificationHomeActivity*

(16) *CreateShortcutActivity*

(17) *CreateDocumentScanShortcutActivity*

(18) *WidgetConfigureActivity*

(19) *ProjectorActivity*

(20) *PdfViewerActivity*

(21) *OpenSafUrlActivity*

(22) *GlobalSearch*

(23) *GetContentActivity*

(24) *PickActivity*

(25) *DocumentOpenerActivityProxy*

(26) *PunchOpenUrlActivityAlias*

(27) *TrixOpenUrlActivityAlias*

(28) *KixOpenUrlActivityAlias*

(29) *DriveOpenUrlActivityAlias*

(30) *com.google.android.gms.actions*

(31) *com.google.android.apps.docs.actions*

(32) *com.google.android.apps.docs*

(33) *com.google.android.apps.docs.notification*

(34) *android.appwidget.action*

(35) *com.google.android.apps.docs.storage*

(36) *docs.google.com*

(37) *application/vnd.google-apps*

(38) One of: *pathPatten + scheme + host =”” + http + docs.google.com*
a/./ + http + docs.google.com*
m + http + docs.google.com
a/./m + http + docs.google.com*
folder. + http + docs.google.com*
a/./folder.* + http + docs.google.com*
file/. + http + docs.google.com*
a/./file/.* + http + docs.google.com*
open + http + docs.google.com
a/./open + http + docs.google.com*
leaf + http + docs.google.com
a/./leaf + http + docs.google.com*
uc + http + docs.google.com
a/./uc + http + docs.google.com*

viewer + http + docs.google.com
a/./viewer + http + docs.google.com*
"" + https + docs.google.com
a/./ + https + docs.google.com*
m + https + docs.google.com
a/./m + https + docs.google.com*
folder. + https + docs.google.com*
a/./folder.* + https + docs.google.com*
file/. + https + docs.google.com*
a/./file/.* + https + docs.google.com*
open + https + docs.google.com
a/./open + https + docs.google.com*
leaf + https + docs.google.com
a/./leaf + https + docs.google.com*
uc + https + docs.google.com
a/./uc + https + docs.google.com*
viewer + https + docs.google.com
a/./viewer + https + docs.google.com*

(39) One of: *pathPatten + scheme + host =/document/.* + http + docs.google.com*
/a/./document/.* + http + docs.google.com*
/Doc + http + docs.google.com
/a/./Doc + http + docs.google.com*
/View + http + docs.google.com
/a/./View + http + docs.google.com*
/document/. + https + docs.google.com*
/a/./document/.* + https + docs.google.com*
/Doc + https + docs.google.com
/a/./Doc + https + docs.google.com*
/View + https + docs.google.com
/a/./View + https + docs.google.com*

(40) One of: *pathPatten + scheme + host =/present/.* + s:http + docs.google.com*
/a/./present/.* + s:http + docs.google.com*
presentation/. + s:http + docs.google.com*
/a/./presentation/.* + s:http + docs.google.com*
/present/. + s:https + docs.google.com*
/a/./present/.* + s:https + docs.google.com*
/presentation/. + s:https + docs.google.com*
/a/./presentation/.* + s:https + docs.google.com*

(41) One of: *pathPatten + scheme + host =/spreadsheets/.* + s:http + docs.google.com*
/spreadsheet/. + s:http + docs.google.com*
/a/./spreadsheet/.* + s:http + docs.google.com*
/spreadsheets/d/. + s:http + docs.google.com*
/a/./spreadsheets/d/.* + s:http + docs.google.com*
/spreadsheets/. + s:https + docs.google.com*
/spreadsheet/. + s:https + docs.google.com*
/a/./spreadsheet/.* + s:https + docs.google.com*
/spreadsheets/d/. + s:https + docs.google.com*
/a/./spreadsheets/d/.* + s:https + docs.google.com*

(42) One of: *scheme + mimeType + host =content + application/vnd.google-apps.drive-sdk.generic + com.google.android.apps.docs.storage*
content + application/vnd.google-apps.drawing + com.google.android.apps.docs.storage
content + application/vnd.google-apps.form + com.google.android.apps.docs.storage
content + application/vnd.google-apps.map + com.google.android.apps.docs.storage
content + application/vnd.google-apps.site + com.google.android.apps.docs.storage
content + application/vnd.google-apps.table + com.google.android.apps.docs.storage

(43) One of: *scheme + host =http + spreadsheets.google.com*
https + h:spreadsheets.google.com

Table A-28. *Duo Activity Intent Filters*

Action	Category	Data	Act
VIEW			RegistrationActivity
VIEW			VerificationActivity
MAIN	LAUNCHER		MainActivity
VIEW	DEFAULT		TagManagerPreviewActivity
	BROWSABLE		

Table A-29. *EasterEgg Activity Intent Filters*

Action	Category	Data	Act
MAIN	DEFAULT		Ocquarium
	(1).PLATLOGO		
(2).QS_TILE_PREFERENCES			NekoLand
MAIN			
MAIN	DEFAULT		NekoActivationActivity

(1) *com.android.internal.category*

(2) *android.service.quicksettings.action*

Table A-30. *ExactCalculator Activity Intent Filters*

Action	Category	Data	Act
MAIN	LAUNCHER APP_CALCULATOR		Calculator

Table A-31. *GoogleHindiIME Activity Intent Filters*

Action	Category	Data	Act
MAIN	LAUNCHER		LauncherActivity
MAIN	LEANBACK_LAUNCHER		LauncherActivity
MAIN			HinglishSpellCheckerSettingsActivity

Table A-32. GooglePinyinIME Activity Intent Filters

Action	Category	Data	Act
MAIN	LAUNCHER		LauncherActivity
MAIN	LEANBACK_LAUNCHER		LauncherActivity
(1).DEBUG_PRIMES_EVENTS	DEFAULT		PrimesEventActivity

(1) *com.google.android.primes.action*

Table A-33. GoogleTTS Activity Intent Filters

Action	Category	Data	Act
(1).CHECK_TTS_DATA	DEFAULT		CheckVoiceData
(1).GET_SAMPLE_TEXT	DEFAULT		GetSampleText
(2).EngineSettings	DEFAULT		EngineSettings
(1).INSTALL_TTS_DATA	DEFAULT		VoiceDataInstallActivity

(1) *android.speech.tts.engine*

(2) *com.google.android.tts.settings*

Table A-34. HTMLViewer Activity Intent Filters

Action	Category	Data	Act
VIEW	DEFAULT	scheme:file	(1)
		scheme:content	
		mimeType:text/html	
		mimeType:text/plain	
		mimeType:application/xhtml+xml	
		mimeType:application/vnd.wap.xhtml+xml	

(1) *HTMLViewerActivity*

Table A-35. KeyChain Activity Intent Filters

Action	Category	Data	Act
(1).CHOOSER	DEFAULT		KeyChainActivity

(1) *com.android.keychain*

Table A-36. LatinIMEGooglePrebuilt Activity Intent Filters

Action	Category	Data	Act
MAIN	LAUNCHER		LauncherActivity
MAIN			LatinSpellCheckerSettingsActivity
(1).DEBUG_PRIMES_EVENTS	DEFAULT		PrimesEventActivity

(1) *com.google.android.primes.action*

Table A-37. LatinIMEGooglePrebuilt Activity Intent Filters

Action	Category	Data	Act
MAIN	LAUNCHER		LauncherActivity
MAIN			LatinSpellCheckerSettingsActivity
(1).DEBUG_PRIMES_EVENTS	DEFAULT		PrimesEventActivity

(1) *com.google.android.primes.action*

Table A-38. LiveWallpapersPicker Activity Intent Filters

Action	Category	Data	Act
(1).LIVE_WALLPAPER_CHOOSER SET_WALLPAPER	DEFAULT		LiveWallpaperActivity
(1).CHANGE_LIVE_WALLPAPER	DEFAULT		LiveWallpaperChange

(1) *android.service.wallpaper*

Table A-39. Maps Activity Intent Filters

Action	Category	Data	Act
VIEW	DEFAULT BROWSABLE	pathPrefix:/oauth2redirect scheme:google.maps.taxi	(1)
VIEW	DEFAULT BROWSABLE	scheme:googlemapstaxi host:oauth2redirect	(1)
MAIN	LAUNCHER APP_MAPS, DEFAULT		(2)
VIEW	DEFAULT BROWSABLE	scheme:peterparker	(2)
VIEW	DEFAULT	scheme:geo.replay	(2)
VIEW	DEFAULT BROWSABLE	scheme:google.navigation	(2)

(*continued*)

Table A-39. (*continued*)

Action	Category	Data	Act
VIEW	DEFAULT BROWSABLE	scheme:geo	(2)
VIEW	DEFAULT	scheme:google.maps	(2)
SEND SEND_MULTIPLE	DEFAULT	mimeType:-application/vnd.google.-panorama360+jpg mimeType:image/*	(2)
VIEW	DEFAULT BROWSABLE	scheme:googlemapstaxi host:addpayment	(2)
VIEW	DEFAULT BROWSABLE	scheme:http scheme:https path:/ (13) (14)	(2)
VIEW	DEFAULT BROWSABLE	scheme:http scheme:https pathPrefix:/map/viewer host:mapsengine.google.com pathPrefix:/map/u/.*/viewer host:mapsengine.google.com	(2)
(15).NDEF_DISCOVERED	DEFAULT	scheme:http scheme:https path:/ (14)	(2)
VIEW	DEFAULT BROWSABLE	scheme:http scheme:https path:/maps path:/maps/preview (16) pathPattern:/maps/d/u/.*/viewer (17)	(2)
VIEW	DEFAULT BROWSABLE	scheme:http scheme:https pathPrefix:/maps host:goo.gl	(2)
(15).NDEF_DISCOVERED	DEFAULT	scheme:google.navigation	(2)

(*continued*)

Table A-39. (*continued*)

Action	Category	Data	Act
VIEW	DEFAULT	mimeType:vnd.android.cursor.item/postal-address_v2	(2)
MANAGE_NETWORK_USAGE	DEFAULT		(2)
MAIN	NOTIFICATION_PREFERENCES		(2)
VIEW	DEFAULT	scheme:google.streetview	(2)
	BROWSABLE		
(18).SEARCH_ACTION	DEFAULT		(2)
(18).VIEW	DEFAULT	scheme:google.maps.timeline	(2)
VIEW	DEFAULT	scheme:http	(2)
	BROWSABLE	scheme:https	
		host:plus.codes	
		(19)	
CREATE_SHORTCUT	DEFAULT		(3)
	CAR_MODE		
CREATE_SHORTCUT	DEFAULT		(4)
CREATE_SHORTCUT	DEFAULT		(5)
CREATE_SHORTCUT	DEFAULT		(6)
CREATE_SHORTCUT	DEFAULT		(7)
(20).PROJECTED_FIRST_RUN	DEFAULT		(8)
(21).ACTION_PREVIEW]	DEFAULT		(9)
MAIN			(10)
MAIN			(11)
VIEW	DEFAULT		(12)

(1) *RedirectUriReceiverActivity*

(2) *MapsActivity*

(3) *CreateDirectionsShortcutActivity*

(4) *FreeNavCreateShortcutActivity*

(5) *TrafficHubCreateShortcutActivity*

(6) *LocationSharingCreateShortcutActivity*

(7) *SelectedPersonCreateShortcutActivity*

(8) *GmmProjectedFirstRunActivity*

(9) *PreviewActivity*

(10) *PlacesActivity*

(11) *DestinationActivity*

(12) *NavigationActivity*

(13) One of *pathPrefix: /locationhistory*, */maps*,*/maps/me*, */localguides/signup*

(14) All relevant Google Map hosts: e.g., *host:maps.google.[...]*

(15) *android.nfc.action*

(16) One of: *pathPrefix:/locationhistory, /maps/timeline/maps/contrib /local/guides/signup/ maps/@, /maps/place//maps/search/, /maps/dir//maps/offline, /maps/placelists/all/maps/ placelists/list/, /maps/preview/@/maps/preview/place/, /maps/preview/search//maps/preview/ dir/, /maps/d/viewer/maps/me*

(17) Alternate set of Google Map hosts: e.g., *host:maps.google.[...]*

(18) *com.google.android.gms.actions*

(19) One of: *pathPrefix: /2, /3, /4, /5, /6, /7, /8, /9, /C, /F, /G, /H, /J, /M/P, /Q, /R, /V, /W, /X*

(20) *com.google.android.apps.maps.car*

(21) *com.google.android.gms.appinvite*

Table A-40. Music2 Activity Intent Filters

Action	Category	Data	Act
(1).PLAY	DEFAULT		(2)
(1).SHARED_PLAY	DEFAULT		(3)
VIEW	DEFAULT, BROWSABLE	pathPrefix:/music/playlist + scheme:https + host:play.google.com pathPrefix:/music/playlist + scheme:http + host:play.google.com	(3)
(1).MUSIC_SETTINGS	DEFAULT		(4)
MAIN	NOTIFICATION_ PREFERENCES		(4)
(1).LICENSES	DEFAULT		(5)
(1).SELECT_ACCOUNT	DEFAULT		(6)
(1).OPEN_WEAR_SYNC_ MANAGEMENT_UI	DEFAULT		(7)

(*continued*)

Table A-40. (*continued*)

Action	Category	Data	Act
VIEW	DEFAULT BROWSABLE	scheme:file mimeType:audio/* mimeType:application/ogg mimeType:application/x-ogg mimeType:application/itunes	(8)
VIEW	DEFAULT BROWSABLE	scheme:http mimeType:audio/* mimeType:application/ogg mimeType:application/x-ogg mimeType:application/itunes	(8)
VIEW	DEFAULT BROWSABLE	scheme:https mimeType:audio/* mimeType:application/ogg mimeType:application/x-ogg mimeType:application/itunes	(8)
VIEW	DEFAULT BROWSABLE	scheme:content mimeType:audio/* mimeType:application/ogg mimeType:application/x-ogg mimeType:application/itunes	(8)
PICK	DEFAULT OPENABLE	mimeType:vnd.android. cursor. dir/audio	(9)
SEARCH	DEFAULT	(10)	
VIEW	DEFAULT	mimeType:vnd.android. cursor. dir/vnd. google.music.playlist mimeType:vnd.android. cursor. dir/playlist	(11)
(1).shortcuts.START_IFL		(12)	
(1).shortcuts.MY_LIBRARY		(12)	
(1).shortcuts.RECENT_ ACTIVITY		(12)	
VIEW	DEFAULT BROWSABLE	scheme:googleplaymusic	(13)
VIEW	DEFAULT BROWSABLE	(24) or (25)	(13)
(22).MEDIA_PLAY_FROM_ SEARCH	DEFAULT		(14)

(*continued*)

Table A-40. (*continued*)

Action	Category	Data	Act
(1).DeviceManagement	DEFAULT		(15)
VIEW	DEFAULT	scheme:playmusic host:(1)	(16)
(23).ACTION_PREVIEW	DEFAULT		(17)
MAIN MUSIC_PLAYER	DEFAULT LAUNCHER APP_ MUSIC		(18)
(1).PLAYBACK_VIEWER	DEFAULT		(18)
VIEW	DEFAULT	mimeType:vnd.android. cursor. item/vnd. google.music.album mimeType:vnd.android. cursor. dir/artistalbum mimeType:vnd. android. cursor.dir/album mimeType:vnd.android. cursor. dir/vnd. google.music.album mimeType:vnd.android. cursor. dir/track	(18)
MANAGE_NETWORK_USAGE	DEFAULT		(19)
CREATE_SHORTCUT	DEFAULT CAR_MODE		(20)
GET_CONTENT	DEFAULT OPENABLE	mimeType:audio/* mimeType:application/ogg mimeType:application/x-ogg	(21)

(1) *com.google.android.music*

(2) *PlaySongsActivity*

(3) *SharedSongsActivity*

(4) *MusicSettingsActivity*

(5) *LicenseActivity*

(6) *TutorialSelectAccountActivity*

(7) *ManageWearDownloadsActivity*

(8) *AudioPreviewActivity*

(9) *MusicPickerActivity*

(10) *ClusteredSearchActivity*

(11) *TrackContainerActivity*

(12) *ShortcutTrampolineActivity*

(13) *MusicUrlHandlerActivity*

(14) *VoiceActionsActivity*

(15) *ManageDevicesActivity*

(16) *AppNavigationTrampolineActivity*

(17) *PreviewActivity*

(18) *TopLevelActivity*

(19) *NetworkUsage*

(20) *PlaylistShortcutActivity*

(21) *GetContent*

(22) *android.media.action*

(23) *com.google.android.gms.appinvite*

(24) One of: *triplets pathPattern + scheme + host:/music/m/T.* + https + play*.google.com*/music/m/T.* + http + play*.google.com*/music/m/B.* + https + play*.google.com*/music/m/B.* + http + play*.google.com*/music/m/A.* + https + play*.google.com*/music/m/A.* + http + play*.google.com*/music/m/L.* + https + play*.google.com*/music/m/L.* + http + play*.google.com*/music/m/I.* + https + play*.google.com*/music/m/I.* + http + play*.google.com*/music/m/D.* + https + play*.google.com*/music/m/D.* + http + play*.google.com*/music/m/N.* + https + play*.google.com*/music/m/N.* + http + play*.google.com*/music/r/m/T.* + https + play*.google.com*/music/r/m/T.* + http + play*.google.com*/music/r/m/B.* + https + play*.google.com*/music/r/m/B.* + http + play*.google.com*/music/r/m/A.* + https + play*.google.com*/music/r/m/A.* + http + play*.google.com*/music/r/m/L.* + https + play*.google.com*/music/r/m/L.* + http + play*.google.com*/music/r/playlist/A.* + https + play*.google.com*/music/r/playlist/A.* + http + play*.google.com*/music/gift + https + play*.google.com*/music/gift + http + play*.google.com*/music/podcasts + https + play*.google.com*/music/podcasts + http + play*.google.com*/music/topcharts + https + play*.google.com*/music/topcharts + http + play*.google.com*/music/newreleases + https + play*.google.com*/music/newreleases + http + play*.google.com*/music/settings + https + play*.google.com*/music/settings + http + play*.google.com*/music/settings/* + https + play*.google.com*/music/settings/* + http + play*.google.com

(25) One of: *scheme + path + host:https + /music/listen + music*.google.com *http + /music/listen + music*.google.com *https + /music/uq + play*.google.com*http + /music/uq + play*.google.com *https + /music + play*.google.com *http + /music + play*.google.com

Table A-41. NetSpeed Activity Intent Filters

Action	Category	Data	Act
MAIN	DEFAULT		NetSpeedActivity

Table A-42. OpenWnn Activity Intent Filters

Action	Category	Data	Act
MAIN			OpenWnnControlPanelJAJP

Table A-43. Photos Activity Intent Filters

Action	Category	Data	Act
(1).CROP	DEFAULT ALTERNATIVE SELECTED_ALTERNATIVE	mimeType:image/*	(2)
EDIT	DEFAULT	mimeType:image/*	(2)
MAIN	DEFAULT LAUNCHER APP_ GALLERY INFO		(3)
VIEW (1).REVIEW	DEFAULT BROWSABLE	scheme:"" scheme:http scheme:https scheme:content scheme:file mimeType:image/* mimeType:application/vnd. google.panorama360+jpg mimeType:video/mpeg mimeType:video/mpeg4 mimeType:video/mp4 mimeType:video/3gp mimeType:video/3gpp mimeType:video/3gpp2 mimeType:video/webm mimeType:video/avi mimeType:video/x-matroska mimeType:video/quicktime mimeType:application/sdp	(4)
VIEW	DEFAULT BROWSABLE	scheme:rtsp	(4)

(*continued*)

Table A-43. (*continued*)

Action	Category	Data	Act
VIEW (1).REVIEW	DEFAULT	scheme:http	(4)
	BROWSABLE	scheme:https	
		mimeType:audio/x-mpegurl	
		mimeType:audio/mpegurl	
		mimeType:application/vnd. apple. mpegurl	
		mimeType:application/xmpegurl	
VIEW	DEFAULT	mimeType:vnd.	(4)
		android.cursor.dir/image	
		mimeType:vnd.android.cursor. dir/ video	
(6).edit	DEFAULT	mimeType:image/*	(5)
PICK	DEFAULT	mimeType:image/*	(7)
		mimeType:video/*	
		mimeType:vnd.android.cursor. dir/ image	
		mimeType:vnd.android.cursor. dir/ video	
GET_CONTENT	OPENABLE DEFAULT	mimeType:image/*	(7)
		mimeType:video/*	
		mimeType:vnd.android.cursor. dir/ image	
(21).NOTIFICATION_ SETTINGS	DEFAULT		(8)
ATTACH_DATA	DEFAULT	mimeType:image/*	(9)
SET_WALLPAPER	DEFAULT		(9)
SEND	DEFAULT	mimeType:application/vnd. google. panorama360+jpg	(10)
		mimeType:image/*	
		mimeType:video/*	
SEND_MULTIPLE	DEFAULT	mimeType:application/vnd. google. panorama360+jpg	(10)
		mimeType:image/*	
		mimeType:video/*	

(*continued*)

Table A-43. (*continued*)

Action	Category	Data	Act
VIEW	DEFAULT	scheme:gplus, path:/apiaryTrace, host:app	(11)
(13).USB_DEVICE_ATTACHED			(2)
MAIN	DEFAULT NOTIFICATION_PREFERENCES		(14)
EDIT	DEFAULT	mimeType:application/vnd. google. panorama360+jpg	(15)
(1).TRIM	DEFAULT	scheme:content scheme:file mimeType:video/*	(15)
VIEW	DEFAULT BROWSABLE	pathPattern:/share/.* + scheme:http + host:photos.google.com pathPattern:/share/.* + scheme:https + host:photos.google.com pathPattern:/u/.*/share/.* + scheme:http + host:photos.google.com pathPattern:/u/.*/share/.* + scheme:https + host:photos.google.com pathPattern:/photos/.* + scheme:http + host:goo.gl pathPattern:/photos/.* + scheme:https + host:goo.gl	(16)

(*continued*)

Table A-43. (*continued*)

Action	Category	Data	Act
VIEW	DEFAULT BROWSABLE	pathPattern:/share/.*/inapp + scheme:http + host:photos.google.com pathPattern:/share/.*/inapp + scheme:https + host:photos.google.com pathPattern:/u/.*/share/.*/inapp + scheme:http + host:photos.google.com pathPattern:/u/.*/share/.*/inapp + scheme:https + host:photos.google.com]]]	(17)
VIEW	DEFAULT BROWSABLE	pathPattern:/assistant + scheme:http + host:photos.google.com pathPattern:/assistant + scheme:https + host:photos.google.com	(18)
(22).SEARCH_ACTION	DEFAULT		(19)
VIEW	DEFAULT BROWSABLE	pathPattern:/search/.* + scheme:http + host:photos.google.com pathPattern:/search/.* + scheme:https + host:photos.google.com pathPattern:/u/.*/search/.* + scheme:http + host:photos.google.com pathPattern:/u/.*/search/.* + scheme:https + host:photos.google.com	(19)
MAIN	DEFAULT NOTIFICATION_ PREFERENCES		(20)

(1) *com.android.camera.action*

(2) *EditActivity*

(3) *HomeActivity*

(4) *HostPhotoPagerActivity*

(5) *ConsumerPhotoEditorActivity*

(6) *com.google.android.apps.photos.editor*

(7) *ExternalPickerActivity*

(8) *SettingsActivity*

(9) *SetWallpaperActivity*

(10) *UploadContentActivity*

(11) *TracingTokenQrCodeActivity*

(12) *IngestActivity*

(13) *android.hardware.usb.action*

(14) *NotificationSettingsActivity*

(15) *EditVideoActivity*

(16) *EnvelopeActivityAlias*

(17) *SharedAlbumPromoActivityAlias*

(18) *HomeActivityAlias*

(19) *SearchActivityAlias*

(20) *NotificationSettingsActivityAlias*

(21) *com.google.android.libraries.social.settings*

(22) *com.google.android.gms.actions*

Table A-44. *PrebuiltBugle Activity Intent Filters*

Action	Category	Data	Act
VIEW SENDTO	DEFAULT BROWSABLE	scheme:sms scheme:smsto	(1)
VIEW SENDTO	DEFAULT BROWSABLE	scheme:mms scheme:mmsto	(1)
MAIN	DEFAULT NOTIFICATION_PREFERENCES		(2)
SEND	DEFAULT	mimeType:text/plain mimeType:text/x-vCard mimeType:text/x-vcard mimeType:image/* mimeType:audio/* mimeType:application / ogg	(3)

(*continued*)

Table A-44. (*continued*)

Action	Category	Data	Act
SEND_MULTIPLE	DEFAULT	mimeType:image/*	(3)
(7).APPWIDGET_CONFIGURE			(4)
MAIN	LAUNCHER DEFAULT APP_MESSAGING		(5)
SEND	DEFAULT	mimeType:video/*	(6)

(1) *LaunchConversationActivity*

(2) *ApplicationSettingsActivity*

(3) *ShareIntentActivity*

(4) *WidgetPickConversationActivity*

(5) *ConversationListActivity*

(6) *VideoShareIntentActivity*

(7) *android.appwidget.action*

Table A-45. PrebuiltDeskClockGoogle Activity Intent Filters

Action	Category	Data	Act
VIEW	DEFAULT	scheme:clock-app host:com.google.android. deskclock	(1)
MAIN	DEFAULT LAUNCHER		(1)
DISMISS_ALARM SHOW_ALARMS SHOW_TIMERS SNOOZE_ALARM	DEFAULT VOICE		(1)
SET_ALARM SET_TIMER	DEFAULT VOICE		(1)

(1) *HandleUris*

(2) *DeskClock*

(3) *HandleApiCalls*

(4) *HandleSetAlarmApiCalls*

Table A-46. PrebuiltGmail Activity Intent Filters

Action	Category	Data	Act
(1).FORCE_CREATE_ACCOUNT (30).CREATE_NEW_ACCOUNT	DEFAULT		(2)
MAIN			(2)
VIEW	DEFAULT	host:com.google.android.gm.email. ACCOUNT_SETTINGS scheme:auth	(3)
VIEW	DEFAULT	host:com.google.android.gm.email. ACCOUNT_SECURITY scheme:auth	(4)
SENDTO VIEW	DEFAULT BROWSABLE	scheme:mailto	(5)
(26).NDEF_DISCOVERED	DEFAULT	scheme:mailto	(5)
SEND	DEFAULT	pathPrefix:/compose scheme:content host:ui.email2.android.com	(5)
SEND	DEFAULT	host:gmail-ls scheme:gmail2from	(5)
SEND	DEFAULT (32).SELF_NOTE	mimeType:*/*	(5)
SEND_MULTIPLE	DEFAULT	mimeType:*/*	(5)
(27).LAUNCH_COMPOSE	DEFAULT		(6)
(27).LAUNCH_COMPOSE	DEFAULT	scheme:content	(6)
(27).GIG_ACTION_REPLY_TO_ITEM_NOTIFICATION		scheme:content	(6)
(28).AUTO_SEND	DEFAULT	mimeType:*/*	(7)
(29).APPWIDGET_CONFIGURE			(8)
CREATE_SHORTCUT	DEFAULT		(9)
(30).ACCOUNT_MANAGER_ENTRY_INTENT	DEFAULT		(10)
MANAGE_NETWORK_USAGE	DEFAULT		(10)
MAIN	DEFAULT NOTIFICATION_PREFERENCES		(10)
VIEW	DEFAULT	host:gmail scheme:auth	(11)

(*continued*)

Table A-46. (*continued*)

Action	Category	Data	Act
VIEW	DEFAULT	scheme:content mimeType:application/ gm-email-ls	(12)
VIEW	DEFAULT	host:com.android.gmail.ui scheme:content path:/proxy	(13)
VIEW	DEFAULT	mimeType:message/rfc822 mimeType:application/eml	(14)
(31).VIEW_PLID	DEFAULT		(15)
	DEFAULT		(16)
SEARCH	DEFAULT		(17)
MAIN	(33).EMAIL		(18)
MAIN	(33).FIRST_ IMPRESSION		(18)
(29).APPWIDGET_CONFIGURE			(19)
CREATE_SHORTCUT	DEFAULT		(20)
EDIT, VIEW	DEFAULT	host:gmail scheme:setting	(21)
EDIT, VIEW	DEFAULT	host:com.google.android.gm scheme:feedback	(21)
EDIT, VIEW	DEFAULT	pathPrefix:/settings scheme:content host:ui.email.android.com	(21)
VIEW	DEFAULT	scheme:content host:com.google.android.gm mimeType:vnd.android. cursor.item/vnd.com. google.android.gm.label	(22)
MAIN			(23)
MAIN	DEFAULT LAUNCHER BROWSABLE APP_ EMAIL		(24)

(*continued*)

Table A-46. (*continued*)

Action	Category	Data	Act
VIEW	DEFAULT	host:gmail-ls scheme:content mimeType:application/ gmail-ls	(24)
VIEW	DEFAULT	scheme:content mimeType:application/ gmail-ls	(24)
MAIN	DEFAULT LAUNCHER BROWSABLE APP_ EMAIL		(25)
VIEW	DEFAULT	host:gmail-ls scheme:content mimeType:application/ gmail-ls	(25)
VIEW	DEFAULT	scheme:content mimeType:application/ gmail-ls	(25)

(1) *com.google.android.gm.email*

(2) *AccountSetupFinalGmail*

(3) *HeadlessAccountSettingsLoader*

(4) *AccountSecurity*

(5) *ComposeActivityGmailExternal*

(6) *ComposeActivityGmail*

(7) *AutoSendActivity*

(8) *MailboxSelectionActivityGmail*

(9) *CreateShortcutActivityGmail*

(10) *PublicPreferenceActivity*

(11) *ReauthenticateActivity*

(12) *MailActivityGmail*

(13) *ViewProxyActivity*

(14) *EmlViewerActivityGmail*

(15) *TrampolineActivity*

(16) *OpenBrowserTrampolineActivity*

(17) *MailActivity*

(18) *AccountSetupFinalGmailSuggestions*

(19) *MailboxSelectionActivity*

(20) *CreateShortcutActivityGoogleMail*

(21) *Gmail2PreferenceActivity*

(22) *PublicGmailActivity*

(23) *GmailActivity*

(24) *ConversationListActivityGoogleMail*

(25) *ConversationListActivityGmail*

(26) *android.nfc.action*

(27) *com.android.mail.intent.action*

(28) *com.google.android.gm.action*

(29) *android.appwidget.action*

(30) *com.google.android.gm.email*

(31) *com.google.android.gm.intent*

(32) *com.google.android.voicesearch*

(33) *com.android.settings.suggested.category*

Table A-47. *PrintSpooler Activity Intent Filters*

Action	Category	Data	Act
(1).PRINT_DIALOG	DEFAULT	pathPattern:* scheme:printjob	(2)

(1) *android.print*

(2) *PrintActivity*

Table A-48. Videos Activity Intent Filters

Action	Category	Data	Act
(1).VIEW	DEFAULT BROWSABLE	scheme:http	(2)
		scheme:https	
		host:youtube.com	
		host:*.youtube.com	
		pathPrefix:/watch	
		pathPrefix:/show	
VIEW (1).VIEW	DEFAULT BROWSABLE	scheme:http	(2)
		scheme:https	
		host:play.google.com	
		pathPrefix:/movies	
MAIN	DEFAULT INFO		(2)
SEARCH	DEFAULT		(4)
(1).REDEEM	DEFAULT		(5)
(3).bugreport	DEFAULT		(6)
(11).VIEW	DEFAULT BROWSABLE	scheme:http	(7)
		scheme:https	
		host:youtube.com	
		host:*.youtube.com	
		pathPrefix:/watch	
(12).VIEW	DEFAULT BROWSABLE	scheme:http	(7)
		scheme:https	
		host:play.google.com	
		pathPrefix:/trailers	
(12).PLAY	DEFAULT	scheme:http	(8)
(12).PIN		scheme:https	
(12).UNPIN		host:play.google.com	
(12).PURCHASE		pathPrefix:/movies/api	
MAIN	DEFAULT LAUNCHER		(9)
MANAGE_NETWORK_USAGE	DEFAULT		(10)

(1) *com.google.android.videos.intent.action*

(2) *LauncherActivity*

(3) *com.google.android.youtube.videos.cast.action*

(4) *SearchActivity*

(5) *ChoosesEntryActivity*

(6) LogCollectorActivity

(7) *TrailerLauncherActivity*

(8) *ApiActivity*

(9) *EntryPoint*

(10) *ManageNetworkUsageActivity*

(11) *com.google.android.videos.intent.action.trailers*

(12) *com.google.android.videos.intent.action*

Table A-49. *WallpaperPickerGooglePrebuilt Activity Intent Filters*

Action	Category	Data	Act
SET_WALLPAPER	DEFAULT		(1)
(4).CROP_AND_SET_WALLPAPER	DEFAULT	mimeType:image/*	(2)
MAIN	LAUNCHER		(3)

(1) *TopLevelPickerActivity*

(2) *StandalonePreviewActivity*

(3) *CategoryPickerActivity*

(4) *android.service.wallpaper*

Table A-50. *WebViewStub Activity Intent Filters*

Action	Category	Data	Act
(1).WEBVIEW_LICENSE	DEFAULT		LicenseActivity

(1) *android.settings*

Table A-51. *YouTube Activity Intent Filters*

Action	Category	Data	Act
SEARCH			(1)
(15).CLOSE_PLAYER	DEFAULT BROWSABLE		(1)
(15).FULL_SCREEN	DEFAULT BROWSABLE		(1)
(15).MINI_SCREEN	DEFAULT BROWSABLE		(1)
(15).NORMAL_SCREEN	DEFAULT BROWSABLE		(1)
(15).PAUSE	DEFAULT BROWSABLE		(1)
(15).PLAY	DEFAULT BROWSABLE		(1)
(15).STOP	DEFAULT BROWSABLE		(1)

(*continued*)

Table A-51. (*continued*)

Action	Category	Data	Act
(15).NEXT	DEFAULT BROWSABLE		(1)
(15).PREVIOUS	DEFAULT BROWSABLE		(1)
(15).SKIP_ADS	DEFAULT BROWSABLE		(1)
(16).CONNECT	DEFAULT BROWSABLE		(1)
(16).DISCONNECT	DEFAULT BROWSABLE		(1)
(15).PLAY_NTH_VIDEO	DEFAULT BROWSABLE		(1)
SEARCH MEDIA_SEARCH	DEFAULT		(2)
(17).UPLOAD	DEFAULT	mimeType:video/*	(3)
(17).INTERNAL_UPLOAD	DEFAULT	mimeType:video/*	(4)
(18).MEDIA_PLAY_FROM_SEARCH	DEFAULT		(5)
MAIN	NOTIFICATION_PREFERENCES]		(6)
(17).CREATE_LIVE_STREAM	DEFAULT		(7)
(19).START	DEFAULT]		(8)
(20).PLAY_STORY	DEFAULT		(9)
(21).bugreport	DEFAULT		(10)
MAIN	DEFAULT LAUNCHER		(11)
(22).search	DEFAULT		(11)
(22).trending	DEFAULT		(11)
(22).subscriptions	DEFAULT		(11)
SEND SEND_MULTIPLE	ALTERNATIVE DEFAULT	mimeType:video/*	(12)
VIEW (23).MEDIA_PLAY_FROM_SEARCH (24).NDEF_DISCOVERED	DEFAULT BROWSABLE	scheme:http scheme:https host:youtube.com host:www.youtube.com host:m.youtube.com host:youtu.be pathPattern:.*	(13)
VIEW (23).MEDIA_PLAY_FROM_SEARCH (24).NDEF_DISCOVERED	DEFAULT BROWSABLE	scheme:vnd.youtube scheme:vnd.youtube.launch	(13)
MANAGE_NETWORK_USAGE	DEFAULT		(14)

(1) *WatchWhileActivity*

(2) *Shell$ResultsActivity*

(3) *Shell$UploadActivity*

(4) *UploadActivity*

(5) *Shell$MediaSearchActivity*

(6) *Shell$SettingsActivity*

(7) *Shell$LiveCreationActivity*

(8) *StandalonePlayerActivity*

(9) *MoxieActivity*

(10) *LogCollectorActivity*

(11) *Shell$HomeActivity*

(12) *UploadIntentHandlingActivity*

(13) *UrlActivity*

(14) *ManageNetworkUsageActivity*

(15) *com.google.android.youtube.voice*

(16) *com.google.android.youtube.mdx.voice*

(17) *com.google.android.youtube.intent.action*

(18) *android.media.action*

(19) *com.google.android.youtube.api.StandalonePlayerActivity*

(20) *com.google.android.spotlightstories*

(21) *com.google.android.youtube.action*

(22) *com.google.android.youtube.action.open*

(23) *android.media.action*

(24) *android.nfc.action*

System Broadcasts

This is an elaborated version of the list from `SDK_INST_DIR/platforms/VERSION/data/broadcast_actions.txt`, describing all the system broadcast actions. It includes some entries that are missing from that list. The string in the heading of each entry is written in the broadcast listener declarations inside `AndroidManifest.xml`; programmatically declared listeners should instead use the class field constants also shown.

What broadcasts can be used for and the broadcast methodology in general are described in depth in Chapter 5.

Many of those broadcasts have adjoint extra data. To fetch them all, you can use `Intent.getExtras()`. Listed are the keys, wherever feasible, as class constants. Also, take a look at the online API documentation of the class in question.

The section is split into the following topics:

- User account
- App widgets
- Bluetooth
- USB
- Power management
- Media
- Packages (apps)
- Network
- Wi-Fi
- Notifications
- Telephony
- TV
- Audio
- Bootup and shutdown
- Security
- Various

User Account

- **android.accounts.LOGIN_ACCOUNTS_CHANGED**

  Constant `LOGIN_ACCOUNTS_CHANGED_ACTION` from class `android.accounts.AccountManager`. Deprecated in API level 26 (Android 8.0); use `addOnAccountsUpdatedListener()` instead. Fires when an account is added or removed or the credentials change.

- **android.accounts.action.ACCOUNT_REMOVED**

- Constant `ACTION_ACCOUNT_REMOVED` from class `android.accounts.AccountManager`. Fires when an account is removed or renamed. Extra data:

  – `AccountManager.KEY_ACCOUNT_TYPE`: The type of the removed account (`string`)

  – `AccountManager.KEY_ACCOUNT_NAME`: The name of the removed account (`string`)

- **android.app.action.ACTION_PASSWORD_CHANGED**

  Constant `ACTION_PASSWORD_CHANGED` from class `android.app.admin.DeviceAdminReceiver`. Fires when the user changes their password.

- **android.app.action.ACTION_PASSWORD_EXPIRING**

  Constant `ACTION_PASSWORD_EXPIRING` from class `android.app.admin.DeviceAdminReceiver`. Fires when the password is expiring.

- **android.app.action.ACTION_PASSWORD_FAILED**

  Constant `ACTION_PASSWORD_FAILED` from class `android.app.admin.DeviceAdminReceiver`. The user failed to enter the correct password.

- **android.app.action.ACTION_PASSWORD_SUCCEEDED**

  Constant `ACTION_PASSWORD_SUCCEEDED` from class `android.app.admin.DeviceAdminReceiver`. The user entered the correct password.

- **android.app.action.DEVICE_ADMIN_DISABLED**

  Constant `ACTION_DEVICE_ADMIN_DISABLED` from class `android.app.admin.DeviceAdminReceiver`. Action sent to a device administrator when the user has disabled it.

- **android.app.action.DEVICE_ADMIN_DISABLE_REQUESTED**

  Constant `ACTION_DEVICE_ADMIN_DISABLE_REQUESTED` from class `android.app.admin.DeviceAdminReceiver`. Action sent to a device administrator when the user has requested to disable it. Extra data:

  - `DeviceAdminReceiver.EXTRA_DISABLE_WARNING`: You can put a warning string here that will then be shown to the user while disabling admin rights (`string`).

- **android.app.action.DEVICE_ADMIN_ENABLED**

  Constant `ACTION_DEVICE_ADMIN_ENABLED` from class `android.app.admin.DeviceAdminReceiver`. The device is being enabled for administration.

- **android.app.action.DEVICE_OWNER_CHANGED**

  Constant `ACTION_DEVICE_OWNER_CHANGED` from class `android.app.admin.DevicePolicyManager`. The device owner is set, changed, or cleared.

- **android.app.action.MANAGED_PROFILE_PROVISIONED**

  Constant `ACTION_MANAGED_PROFILE_PROVISIONED` from class `android.app.admin.DevicePolicyManager`. Sent to indicate that provisioning of a managed profile has completed successfully.

- Contains extra data:

  - `android.content.Intent.EXTRA_USER`: The user (`android.os.UserHandle`)

  - `android.content.Intent.EXTRA_PROVISIONING_ACCOUNT_TO_MIGRATE`: The account (`android.accounts.Account`).

- **android.app.action.PROFILE_PROVISIONING_COMPLETE**

  Constant `ACTION_PROFILE_PROVISIONING_COMPLETE` from class `android.app.admin.DeviceAdminReceiver`. The provisioning of a managed profile or managed device has completed successfully.

- **android.intent.action.MANAGED_PROFILE_AVAILABLE**

  Constant `ACTION_MANAGED_PROFILE_AVAILABLE` from class `android.content.Intent`. A managed profile is now available. Extra data:

  - `intent.EXTRA_USER`: The corresponding user (`android.os.UserHandle`)
  - `intent.EXTRA_QUIET_MODE` (optional): The quiet mode has been switched on or off (Boolean)

- **android.intent.action.MANAGED_PROFILE_REMOVED**

  Constant `ACTION_MANAGED_PROFILE_REMOVED` from class `android.content.Intent`. A managed profile has been removed. Extra data:

  - `intent.EXTRA_USER`: The corresponding user (`android.os.UserHandle`)

- **android.intent.action.MANAGED_PROFILE_UNAVAILABLE**

  Constant `ACTION_MANAGED_PROFILE_UNAVAILABLE` from class

  `android.content.Intent`. A managed profile has become unavailable. Extra data:

  - `intent.EXTRA_USER`: The corresponding user (`android.os.UserHandle`).
  - `intent.EXTRA_QUIET_MODE` (optional): The quiet mode has been switched on or off (Boolean).

- **android.intent.action.MANAGED_PROFILE_UNLOCKED**

  Constant `ACTION_MANAGED_PROFILE_UNLOCKED` from class `android.content.Intent`. A managed profile has been unlocked. Extra data:

  - `intent.EXTRA_USER`: The corresponding user (`android.os.UserHandle`)

- **android.intent.action.UID_REMOVED**

  Constant `ACTION_UID_REMOVED` from class `android.content.Intent`. A user ID has been removed. Extra data:

  - `Intent.EXTRA_UID`: The UID (`int`)

- **android.intent.action.USER_BACKGROUND**

  Constant `ACTION_USER_BACKGROUND` from class `android.content.Intent`. Sent when a user switch is happening to the user going to the background.

- **android.intent.action.USER_FOREGROUND**

  Constant ACTION_USER_FOREGROUND from class android.content.Intent. Sent when a user switch is happening to the user going to the foreground.

- **android.intent.action.USER_INITIALIZE**

  Constant ACTION_USER_INITIALIZE from class android.content.Intent. Sent the first time a user is starting, before ACTION_BOOT_COMPLETED.

- **android.intent.action.USER_PRESENT**

  Constant ACTION_USER_PRESENT from class android.content.Intent. Sent when the user is present after a device wakeup.

- **android.intent.action.USER_UNLOCKED**

  Constant ACTION_USER_UNLOCKED from class android.content.Intent. Sent when a user gets unlocked, now with access to the private encrypted data store.

App Widgets

- **android.appwidget.action.APPWIDGET_DELETED**

  Constant ACTION_APPWIDGET_DELETED from class android.appwidget.AppWidgetManager. An instance of an AppWidget is deleted from its host.

- **android.appwidget.action.APPWIDGET_CONFIGURE**

  Constant ACTION_APPWIDGET_CONFIGURE from class android.appwidget.AppWidgetManager. Sent when it is time to configure your AppWidget while it is being added to a host. Extra data:

  – AppWidgetManager.EXTRA_APPWIDGET_ID: The AppWidget ID to configure (int)

- **android.appwidget.action.APPWIDGET_DISABLED**

  Constant ACTION_APPWIDGET_DISABLED from class android.appwidget.AppWidgetManager. The last AppWidget of this provider is removed from the last host.

- **android.appwidget.action.APPWIDGET_ENABLED**

  Constant ACTION_APPWIDGET_ENABLED from class android.appwidget.AppWidgetManager. An instance of an AppWidget is added to a host for the first time.

- **android.appwidget.action.APPWIDGET_HOST_RESTORED**

  Constant ACTION_APPWIDGET_HOST_RESTORED from class android.appwidget.AppWidgetManager. The AppWidget state related to the host has been restored from backup. Contains extra data:

  - AppWidgetManager.EXTRA_APPWIDGET_OLD_IDS: Old AppWidget IDs (int[])
  - AppWidgetManager.EXTRA_APPWIDGET_IDS: New AppWidget IDs (int[])
  - AppWidgetManager.EXTRA_HOST_ID: The host ID (int)

- **android.appwidget.action.APPWIDGET_RESTORED**

  Constant ACTION_APPWIDGET_RESTORED from class android.appwidget.AppWidgetManager.

  The AppWidget state related to that provider has been restored from backup. Extra data:

  - AppWidgetManager.EXTRA_APPWIDGET_OLD_IDS: Old restored AppWidget IDs (int[])
  - AppWidgetManager.EXTRA_APPWIDGET_IDS: New AppWidget IDs after restoring (int[])

- **android.appwidget.action.APPWIDGET_UPDATE**

  Constant ACTION_APPWIDGET_UPDATE from class android.appwidget.AppWidgetManager. Sent when it is time to update your AppWidget. Extra data:

  - AppWidgetManager.EXTRA_APPWIDGET_IDS: The AppWidget IDs to update

- **android.appwidget.action.APPWIDGET_UPDATE_OPTIONS**

  Constant ACTION_APPWIDGET_OPTIONS_CHANGED from class android.appwidget.AppWidgetManager. Sent when the custom extras for an AppWidget change.

Bluetooth

- **android.bluetooth.a2dp.profile.action.CONNECTION_STATE_CHANGED**

  Constant ACTION_CONNECTION_STATE_CHANGED from class android.bluetooth.BluetoothA2dp. A change in the connection state of the A2DP profile occurred. Contains extra data:

  - android.bluetooth.BluetoothProfile.EXTRA_STATE: The current state of the profile. One of:

    android.bluetooth.BluetoothProfile.STATE_CONNECTED

    android.bluetooth.BluetoothProfile.STATE_DISCONNECTED

    android.bluetooth.BluetoothProfile.STATE_CONNECTING

    android.bluetooth.BluetoothProfile.STATE_DISCONNECTING

- android.bluetooth.BluetoothProfile.EXTRA_PREVIOUS_STATE: The previous state of the profile. The possible values are the same as for EXTRA_STATE.
- android.bluetooth.BluetoothDevice.EXTRA_DEVICE: The remote device (android.bluetooth.BluetoothDevice).

Requires permission android.permission.BLUETOOTH.

- **android.bluetooth.a2dp.profile.action.PLAYING_STATE_CHANGED**

  Constant ACTION_PLAYING_STATE_CHANGED from class android.bluetooth.BluetoothA2dp. A change in the playing state of the A2DP profile occurred. Contains extra data:

  - android.bluetooth.BluetoothProfile.EXTRA_STATE: The current state of the profile. One of:

    android.bluetooth.BluetoothA2dp.STATE_PLAYING

    android.bluetooth.BluetoothA2dp.STATE_NOT_PLAYING
  - android.bluetooth.BluetoothProfile.EXTRA_PREVIOUS_STATE: The previous state of the profile. The possible values are the same as for EXTRA_STATE.
  - android.bluetooth.BluetoothDevice.EXTRA_DEVICE: The remote device (android.bluetooth.BluetoothDevice).

  Requires permission android.permission.BLUETOOTH.

- **android.bluetooth.adapter.action.CONNECTION_STATE_CHANGED**

  Constant ACTION_CONNECTION_STATE_CHANGED from class android.bluetooth.BluetoothAdapter. A change in the connection state of the local Bluetooth adapter to a profile of the remote device occurred. Contains extra data:

  - android.bluetooth.BluetoothAdapter.EXTRA_CONNECTION_STATE: The current state of the profile. One of:

    android.bluetooth.BluetoothProfile.STATE_CONNECTED

    android.bluetooth.BluetoothProfile.STATE_DISCONNECTED

    android.bluetooth.BluetoothProfile.STATE_CONNECTING

    android.bluetooth.BluetoothProfile.STATE_DISCONNECTING
  - android.bluetooth.BluetoothAdapter.EXTRA_PREVIOUS_CONNECTION_STATE: The previous state of the profile. The possible values are the same as for EXTRA_CONNECTION_STATE.
  - android.bluetooth.BluetoothDevice.EXTRA_DEVICE: The remote device (android.bluetooth.BluetoothDevice).

  Requires permission android.permission.BLUETOOTH.

- **android.bluetooth.adapter.action.DISCOVERY_FINISHED**

  Constant `ACTION_DISCOVERY_FINISHED` from class `android.bluetooth.BluetoothAdapter`.

  The local Bluetooth adapter has finished the device discovery process. Requires permission `android.permission.BLUETOOTH`.

- **android.bluetooth.adapter.action.DISCOVERY_STARTED**

  Constant `ACTION_DISCOVERY_STARTED` from class `android.bluetooth.BluetoothAdapter`.

  The local Bluetooth adapter has started the device discovery process. Requires permission `android.permission.BLUETOOTH`.

- **android.bluetooth.adapter.action.LOCAL_NAME_CHANGED**

  Constant `ACTION_LOCAL_NAME_CHANGED` from class `android.bluetooth.BluetoothAdapter`. The adapter has changed its externally visible Bluetooth name. Extra data:

  – `BluetoothAdapter.EXTRA_LOCAL_NAME`: The name (`string`)

  Requires permission `android.permission.BLUETOOTH`.

- **android.bluetooth.adapter.action.SCAN_MODE_CHANGED**

  Constant `ACTION_SCAN_MODE_CHANGED` from class `android.bluetooth.BluetoothAdapter`.

  The Bluetooth scan mode of the local adapter has changed. Contains extra data:

  – `BluetoothAdapter.EXTRA_SCAN_MODE`: The new scan mode.

    Possible values are as follows:

    `BluetoothAdapter.SCAN_MODE_NONE`: Both inquiry and page scan disabled.

    `BluetoothAdapter.SCAN_MODE_CONNECTABLE`: Inquiry scan disabled; page scan enabled.

    `BluetoothAdapter.SCAN_MODE_CONNECTABLE_DISCOVERABLE`: Both inquiry and page scan enabled.

  – `BluetoothAdapter.EXTRA_PREVIOUS_SCAN_MODE`: The old scan mode. The possible values are the same as for `EXTRA_SCAN_MODE`.

  Requires permission `android.permission.BLUETOOTH`.

- **android.bluetooth.adapter.action.STATE_CHANGED**

  Constant `ACTION_STATE_CHANGED` from class `android.bluetooth.BluetoothAdapter`. The state of the local Bluetooth adapter has been changed. Contains extra data:

- `BluetoothAdapter.EXTRA_STATE`: The new state. Possible values are as follows:

  `BluetoothAdapter.STATE_OFF`

  `BluetoothAdapter.STATE_TURNING_ON`

  `BluetoothAdapter.STATE_ON`

  `BluetoothAdapter.STATE_TURNING_OFF`

- `BluetoothAdapter.EXTRA_PREVIOUS_STATE`: The old state. The possible values are the same as for `EXTRA_STATE`.

Requires permission `android.permission.BLUETOOTH`.

- **android.bluetooth.device.action.ACL_CONNECTED**

  Constant `ACTION_ACL_CONNECTED` from class `android.bluetooth.BluetoothDevice`. A low-level (ACL) connection has been established with a remote device. Contains extra data:

  - `BluetoothDevice.EXTRA_DEVICE`: The remote device (`android.bluetooth.BluetoothDevice`).

  Requires permission `android.permission.BLUETOOTH`.

- **android.bluetooth.device.action.ACL_DISCONNECTED**

  Constant `ACTION_ACL_DISCONNECTED` from class `android.bluetooth.BluetoothDevice`.

  A low-level (ACL) disconnection happened with a remote device. Contains extra data:

  - `BluetoothDevice.EXTRA_DEVICE`: The remote device (`android.bluetooth.BluetoothDevice`).

  Requires permission `android.permission.BLUETOOTH`.

- **android.bluetooth.device.action.ACL_DISCONNECT_REQUESTED**

  Constant `ACTION_ACL_DISCONNECT_REQUESTED` from class `android.bluetooth.BluetoothDevice`.

  A low-level (ACL) disconnection with a remote device has been requested. Contains extra data:

  - `BluetoothDevice.EXTRA_DEVICE`: The remote device (`android.bluetooth.BluetoothDevice`).

  Requires permission `android.permission.BLUETOOTH`.

- **android.bluetooth.device.action.BOND_STATE_CHANGED**

  Constant `ACTION_BOND_STATE_CHANGED` from class `android.bluetooth.BluetoothDevice`.

A change in the bond state of a remote device happened. For example, a device is bonded (paired). Contains extra data:

- `BluetoothDevice.EXTRA_DEVICE`: The remote device (`android.bluetooth.BluetoothDevice`).
- `BluetoothDevice.EXTRA_BOND_STATE`: The new bond state.

  Possible values are as follows:

  `BluetoothDevice.BOND_NONE`

  `BluetoothDevice.BOND_BONDING`

  `BluetoothDevice.BOND_BONDED.`
- `BluetoothDevice.EXTRA_PREVIOUS_BOND_STATE`: The previous bond state. The possible values are the same as for `EXTRA_BOND_STATE`.

Requires permission `android.permission.BLUETOOTH`.

- **android.bluetooth.device.action.CLASS_CHANGED**

  Constant `ACTION_CLASS_CHANGED` from class `android.bluetooth.BluetoothDevice`.

  The Bluetooth class of a remote device has changed. Contains extra data:

  - `BluetoothDevice.EXTRA_DEVICE`: The remote device (`android.bluetooth.BluetoothDevice`).
  - `BluetoothDevice.EXTRA_CLASS`: The device class (`android. bluetooth.BluetoothClass`).

  Requires permission `android.permission.BLUETOOTH`.

- **android.bluetooth.device.action.FOUND**

  Constant `ACTION_FOUND` from class `android.bluetooth.BluetoothDevice`. A remote device has been discovered. Contains extra data:

  - `BluetoothDevice.EXTRA_DEVICE`: The remote device (`android.bluetooth.BluetoothDevice`).
  - `BluetoothDevice.EXTRA_CLASS`: The device class (`android. bluetooth.BluetoothClass`).
  - `BluetoothDevice.EXTRA_NAME` (optional): The device-friendly name (`string`).
  - `BluetoothDevice.EXTRA_RSSI` (optional): The device signal strength RSSI (short int).

  Requires permission `android.permission.BLUETOOTH`.

- **android.bluetooth.device.action.NAME_CHANGED**

  Constant `ACTION_NAME_CHANGED` from class `android.bluetooth.BluetoothDevice`.

The externally visible friendly name of a device changed. Contains extra data:

- `BluetoothDevice.EXTRA_DEVICE`: The remote device (`android.bluetooth.BluetoothDevice`).
- `BluetoothDevice.EXTRA_NAME`: The device externalized friendly name (`string`).

Requires permission `android.permission.BLUETOOTH`.

- **android.bluetooth.device.action.PAIRING_REQUEST**

  Constant `ACTION_PAIRING_REQUEST` from class `android.bluetooth.BluetoothDevice`. Indicates a pairing request. Requires permission `android.permission.BLUETOOTH`.

- **android.bluetooth.device.action.UUID**

  Constant `ACTION_UUID` from class `android.bluetooth.BluetoothDevice`. The UUID as a result of a service discovery has been fetched. Contains extra data:

  - `BluetoothDevice.EXTRA_DEVICE`: The remote device (`android.bluetooth.BluetoothDevice`).
  - `BluetoothDevice.EXTRA_UUID`: The UUID (`android.os.ParcelUuid`).

  Requires permission `android.permission.BLUETOOTH`.

- **android.bluetooth.devicepicker.action.DEVICE_SELECTED**

  Fires when one Bluetooth device is selected from the Bluetooth device picker screen. Extra data:

  - `BluetoothDevice.EXTRA_DEVICE`: The remote device (`android.bluetooth.BluetoothDevice`).

  Requires permission `android.permission.BLUETOOTH`.

- **android.bluetooth.devicepicker.action.LAUNCH**

  Someone wants to select one Bluetooth device from the devices list. Extra data:

  - `BluetoothDevice.EXTRA_NEED_AUTH`: Whether authentication is required (boolean)
  - `BluetoothDevice.EXTRA_FILTER_TYPE`: The filter type (an `int`)
  - `BluetoothDevice.EXTRA_LAUNCH_PACKAGE`: Where this Intent comes from (`string`)
  - `BluetoothDevice.EXTRA_LAUNCH_CLASS`: The intent's originating class (`string`)

- **android.bluetooth.headset.action.VENDOR_SPECIFIC_HEADSET_EVENT**

  Constant ACTION_VENDOR_SPECIFIC_HEADSET_EVENT from class android.bluetooth.BluetoothHeadset. The headset has posted a vendor-specific event. Extra data:

  - android.bluetooth.BluetoothDevice.EXTRA_DEVICE: The remote device (android.bluetooth.BluetoothDevice).
  - BluetoothHeadset.EXTRA_VENDOR_SPECIFIC_HEADSET_EVENT_CMD: The vendor-specific command (string)
  - BluetoothHeadset.EXTRA_VENDOR_SPECIFIC_HEADSET_EVENT_CMD_TYPE: The AT command type. One of:

    BluetoothHeadset.AT_CMD_TYPE_READ

    BluetoothHeadset.AT_CMD_TYPE_TEST

    BluetoothHeadset.AT_CMD_TYPE_SET

    BluetoothHeadset.AT_CMD_TYPE_BASIC

    BluetoothHeadset.AT_CMD_TYPE_ACTION.

  - BluetoothHeadset.EXTRA_VENDOR_SPECIFIC_HEADSET_EVENT_ARGS: Command arguments (string array)

The intent has a special category value: the Company ID of the vendor defining the vendor-specific command (int). Requires permission android.permission.BLUETOOTH.

- **android.bluetooth.headset.profile.action.AUDIO_STATE_CHANGED**

  Constant ACTION_AUDIO_STATE_CHANGED from class android.bluetooth.BluetoothHeadset. A change in the audio connection state of the A2DP profile occurred. Contains extra data:

  - android.bluetooth.BluetoothDevice.EXTRA_DEVICE: The remote device (android.bluetooth.BluetoothDevice).
  - BluetoothHeadset.EXTRA_STATE: The new state. The possible values are as follows:

    BluetoothHeadset.STATE_AUDIO_CONNECTED

    BluetoothHeadset.STATE_AUDIO_DISCONNECTED

  - BluetoothHeadset.EXTRA_PREVIOUS_STATE: The old state. The possible values are the same as for EXTRA_STATE.

  Requires permission android.permission.BLUETOOTH.

- **android.bluetooth.headset.profile.action.CONNECTION_STATE_CHANGED**

  Constant ACTION_CONNECTION_STATE_CHANGED from class android.bluetooth.BluetoothHeadset. A change in connection state of the headset profile occurred. Contains extra data:

- android.bluetooth.BluetoothDevice.EXTRA_DEVICE: The remote device (android.bluetooth.BluetoothDevice).
- BluetoothHeadset.EXTRA_STATE: The new state. Possible values are as follows:

  BluetoothHeadset.STATE_CONNECTED

  BluetoothHeadset.STATE_DICONNECTED

  BluetoothHeadset.STATE_CONNECTING

  BluetoothHeadset.STATE_DICONNECTING
- BluetoothHeadset.EXTRA_PREVIOUS_STATE: The old state. The possible values are the same as for EXTRA_STATE.

Requires permission android.permission.BLUETOOTH.

- **android.bluetooth.input.profile.action.CONNECTION_STATE_CHANGED**

  Used to broadcast the change in connection state of the input device profile. Extra data:

  - android.bluetooth.BluetoothProfile.EXTRA_STATE: Current state.

    Possible values are as follows:

    android.bluetooth.BluetoothProfile.STATE_CONNECTED

    android.bluetooth.BluetoothProfile.STATE_DICONNECTED

    android.bluetooth.BluetoothProfile.STATE_CONNECTING

    android.bluetooth.BluetoothProfile.STATE_DICONNECTING
  - android.bluetooth.BluetoothProfile.EXTRA_PREVIOUS_STATE: Previous state.
  - android.bluetooth.BluetoothProfile.EXTRA_DEVICE: The device.

  Requires permission android.permission.BLUETOOTH.

- **android.bluetooth.input.profile.action.PROTOCOL_MODE_CHANGED**

  Undocumented.

- **android.bluetooth.input.profile.action.VIRTUAL_UNPLUG_STATUS**

  Undocumented.

- **android.bluetooth.inputhost.profile.action.CONNECTION_STATE_CHANGED**

  Undocumented.

- **android.bluetooth.pan.profile.action.CONNECTION_STATE_CHANGED**

  Intent used to broadcast the change in connection state of the Pan profile. Extra data:

  - `android.bluetooth.BluetoothPan.EXTRA_STATE`: Current state. One of:

    `BluetoothPan.STATE_DISCONNECTED`

    `BluetoothPan.STATE_CONNECTING`

    `BluetoothPan.STATE_CONNECTED`

    `BluetoothPan.STATE_DISCONNECTING`

  - `android.bluetooth.BluetoothPan.EXTRA_PREVIOUS_STATE`: The previous state. One of:

    `BluetoothPan.STATE_DISCONNECTED`

    `BluetoothPan.STATE_CONNECTING`

    `BluetoothPan.STATE_CONNECTED`

    `BluetoothPan.STATE_DISCONNECTING`

  - `android.bluetooth.BluetoothDevice.EXTRA_DEVICE`: The device.

  - `android.bluetooth.BluetoothPan.EXTRA_LOCAL_ROLE`: The local role the remote device is bound to. One of:

    `BluetoothPan.LOCAL_NAP_ROLE`

    `BluetoothPan.LOCAL_PANU_ROLE`

  Requires `android.Manifest.permission.BLUETOOTH` permission.

USB

- **android.hardware.usb.action.USB_ACCESSORY_ATTACHED**

  Constant `ACTION_USB_ACCESSORY_ATTACHED` from class `android.hardware.usb.UsbManager`. The user attaches an USB accessory. Extra data:

  - `UsbManager.EXTRA_ACCESSORY`: Contains the `UsbAccessory` for the attached accessory (`android.hardware.usb.UsbAccessory`).

- **android.hardware.usb.action.USB_ACCESSORY_DETACHED**

  Constant `ACTION_USB_ACCESSORY_DETACHED` from class `android.hardware.usb.UsbManager`. The user detaches a USB accessory. Extra data:

  - `UsbManager.EXTRA_ACCESSORY`: Contains the `UsbAccessory` for the detached accessory (`android.hardware.usb.UsbAccessory`).

- **android.hardware.usb.action.USB_DEVICE_ATTACHED**

  Constant `USB_DEVICE_ATTACHED` from class `android.hardware.usb.UsbManager`. The user attaches an USB device. Extra data:

  – `UsbManager.EXTRA_DEVICE`: Contains the attached device (`android.hardware.usb.UsbDevice`).

- **android.hardware.usb.action.USB_DEVICE_DETACHED**

  Constant `USB_DEVICE_DETACHED` from class `android.hardware.usb.UsbManager`. The user detaches an USB device. Extra data:

  – `UsbManager.EXTRA_DEVICE`: Contains the detached device (`android.hardware.usb.UsbDevice`).

- **android.intent.action.UMS_CONNECTED**

  Constant `ACTION_UMS_CONNECTED` from class `android.content.Intent`. Deprecated in API level 14. USB mass storage connected. Do not use (instead use `android.os.storage.StorageEventListener`).

- **android.intent.action.UMS_DISCONNECTED**

  Constant `ACTION_UMS_DISCONNECTED` from class `android.content.Intent`. Deprecated in API level 14. USB mass storage disconnected. Do not use (instead use `android.os.storage.StorageEventListener`).

Power Management

- **android.intent.action.ACTION_POWER_CONNECTED**

  Constant `ACTION_POWER_CONNECTED` from class `android.content.Intent`. The device got the power connected.

- **android.intent.action.ACTION_POWER_DISCONNECTED**

  Constant `ACTION_POWER_DISCONNECTED` from class `android.content.Intent`. The device got the power disconnected.

- **android.intent.action.BATTERY_CHANGED**

  Constant `ACTION_BATTERY_CHANGED` from class `android.content.Intent`. Provide battery status changes.

  Extra data:

  – `android.os.BatteryManager.EXTRA_HEALTH`: An `int` containing the current health status. One of:

  `BatteryManager.BATTERY_HEALTH_COLD`

  `BatteryManager.BATTERY_HEALTH_DEAD`

  `BatteryManager.BATTERY_HEALTH_GOOD`

  `BatteryManager.BATTERY_HEALTH_OVERHEAT`

BatteryManager.BATTERY_HEALTH_OVER_VOLTAGE

BatteryManager.BATTERY_HEALTH_UNKNOWN

BatteryManager.BATTERY_HEALTH_UNSPECIFIED_FAILURE

- android.os.BatteryManager.EXTRA_ICON_SMALL: A resource ID of an icon indicating the battery health state
- android.os.BatteryManager.EXTRA_LEVEL: An int containing the battery level, from 0 to BatteryManager.EXTRA_SCALE
- android.os.BatteryManager.EXTRA_PLUGGED: 0 means battery; other ints mean different types of power source
- android.os.BatteryManager.EXTRA_PRESENT: Boolean, whether a battery is present
- android.os.BatteryManager.EXTRA_SCALE: An int containing the maximum battery level
- android.os.BatteryManager.EXTRA_STATUS: One of:

  BatteryManager.BATTERY_STATUS_CHARGING

  BatteryManager.BATTERY_STATUS_DISCHARGING

  BatteryManager.BATTERY_STATUS_FULL

  BatteryManager.BATTERY_STATUS_NOT_CHARGING

  BatteryManager.BATTERY_STATUS_UNKNOWN
- android.os.BatteryManager.EXTRA_TECHNOLOGY: A string describing the battery's technology
- android.os.BatteryManager.EXTRA_TEMPERATURE: An int with the current battery temperature
- android.os.BatteryManager.EXTRA_VOLTAGE: An int with the current battery voltage level

- **android.intent.action.BATTERY_LOW**

  Constant ACTION_BATTERY_LOW from class android.content.Intent. Low battery condition.

- **android.intent.action.BATTERY_OKAY**

  Constant ACTION_BATTERY_OKAY from class android.content.Intent. Battery now OK from formerly being low.

- **android.os.action.DEVICE_IDLE_MODE_CHANGED**

  Constant ACTION_DEVICE_IDLE_MODE_CHANGED from class android.os.PowerManager. Fires when the outcome of PowerManager.isDeviceIdleMode() changes.

- **android.os.action.POWER_SAVE_MODE_CHANGED**

  Constant `ACTION_POWER_SAVE_MODE_CHANGED` from class `android.os.PowerManager`. Fires when the outcome of `PowerManager.isPowerSaveMode()` changes.

Media

- **android.intent.action.MEDIA_BAD_REMOVAL**

  Constant `ACTION_MEDIA_BAD_REMOVAL` from class `android.content.Intent`. A medium (external SD card) was removed, but not previously unmounted properly.

- **android.intent.action.MEDIA_BUTTON**

  Constant `ACTION_MEDIA_BUTTON` from class `android.content.Intent`. The media button has been pressed. Contains extra data:

  - `intent.EXTRA_KEY_EVENT`: The corresponding key event (`android.view.KeyEvent`)

- **android.intent.action.MEDIA_CHECKING**

  Constant `ACTION_MEDIA_CHECKING` from class `android.content.Intent`. An external medium is present and is being disk-checked.

- **android.intent.action.MEDIA_EJECT**

  Constant `ACTION_MEDIA_EJECT` from class `android.content.Intent`. An external medium is *about* to be ejected. The app should now do a cleanup of its data. The mount point is contained in the `Intent.mData` field.

- **android.intent.action.MEDIA_MOUNTED**

  Constant `ACTION_MEDIA_MOUNTED` from class `android.content.Intent`. An external medium has been mounted. Extra data:

  - `read-only`: Whether the medium is mounted read-only (Boolean)

- **android.intent.action.MEDIA_NOFS**

  Constant `ACTION_MEDIA_NOFS` from class `android.content.Intent`. An external medium is present, but it does not contain a valid file system.

- **android.intent.action.MEDIA_REMOVED**

  Constant `ACTION_MEDIA_REMOVED` from class `android.content.Intent`. An external medium has been removed. The mount point is contained in the `Intent.mData` field.

- **android.intent.action.MEDIA_SCANNER_FINISHED**

  Constant `ACTION_MEDIA_SCANNER_FINISHED` from class `android.content.Intent`. The media scanner has finished scanning a directory. The path is contained in the `Intent.mData` field.

- **android.intent.action.MEDIA_SCANNER_SCAN_FILE**

  Constant `ACTION_MEDIA_SCANNER_SCAN_FILE` from class `android.content.Intent`. Request the media scanner to scan a file and add it to its database. The file path must be entered in the `Intent.mData` field.

- **android.intent.action.MEDIA_SCANNER_STARTED**

  Constant `ACTION_MEDIA_SCANNER_STARTED` from class `android.content.Intent`. The media scanner has started scanning a directory. The path is contained in the `Intent.mData` field.

- **android.intent.action.MEDIA_SHARED**

  Constant `ACTION_MEDIA_SHARED` from class `android.content.Intent`. An external medium is unmounted because it is being shared via USB mass storage. The path to the mount point is contained in the `Intent.mData` field.

- **android.intent.action.MEDIA_UNMOUNTABLE**

  Constant `ACTION_MEDIA_UNMOUNTABLE` from class `android.content.Intent`. An external medium is present, but it cannot be mounted. The path to the requested mount point is contained in the `Intent.mData` field.

- **android.intent.action.MEDIA_UNMOUNTED**

  Constant `ACTION_MEDIA_UNMOUNTED` from class `android.content.Intent`. An external medium has been unmounted. The path to the mount point is contained in the `Intent.mData` field.

Packages (Apps)

- **android.intent.action.MY_PACKAGE_REPLACED**

  Constant `ACTION_MY_PACKAGE_REPLACED` from class `android.content.Intent`. A new version of an app is replacing an existing one. Sent only to the app in question.

- **android.intent.action.PACKAGES_SUSPENDED**

  Constant `ACTION_PACKAGES_SUSPENDED` from class `android.content.Intent`. Apps (packages) have been suspended. Extra data:

  - `Intent.EXTRA_CHANGED_PACKAGE_LIST`: The list of apps (`string` array)

- **android.intent.action.PACKAGES_UNSUSPENDED**

  Constant `ACTION_PACKAGES_UNSUSPENDED` from class `android.content.Intent`. Apps (packages) have been unsuspended. Extra data:

  - `Intent.EXTRA_CHANGED_PACKAGE_LIST`: The list of apps (`string` array)

- **android.intent.action.PACKAGE_ADDED**

  Constant `ACTION_PACKAGE_ADDED` from class `android.content.Intent`.

An app was installed on the device. Extra data:

- `Intent.EXTRA_UID`: The UID of the app (`int`)
- `Intent.EXTRA_REPLACING`: If the app was replaced, follows an `ACTION_PACKAGE_REMOVED` (boolean)

- **android.intent.action.PACKAGE_CHANGED**
- Constant `ACTION_PACKAGE_CHANGED` from class `android.content.Intent`. An existing app has been changed. Extra data:
  - `Intent.EXTRA_UID`: The UID of the app (`int`)
  - `Intent.EXTRA_CHANGED_COMPONENT_NAME_LIST`: A list of changed apps or components (`string` array)
- **android.intent.action.PACKAGE_DATA_CLEARED**

  Constant `ACTION_PACKAGE_DATA_CLEARED` from class `android.content.Intent`. The data of an app have been cleared. Extra data:
  - `Intent.EXTRA_UID`: The UID of the app; will be -1 if this is an *Instant app* (`int`)
  - `Intent.EXTRA_PACKAGE_NAME`: If the app is an *instant app*, the package name (`string`)
- **android.intent.action.PACKAGE_FIRST_LAUNCH**

  Constant `ACTION_PACKAGE_FIRST_LAUNCH` from class `android.content.Intent`. An app was launched the first time. The URI from `Intent.getData()`contains the package name.
- **android.intent.action.PACKAGE_FULLY_REMOVED**

  Constant `ACTION_PACKAGE_FULLY_REMOVED` from class `android.content.Intent`. An app including its data has been removed from the system. Extra data:
  - `Intent.EXTRA_UID`: The UID of the app (`int`)
- **android.intent.action.PACKAGE_INSTALL**

  Constant `ACTION_PACKAGE_INSTALL` from class `android.content.Intent`. Defunct; do not use.
- **android.intent.action.PACKAGE_NEEDS_VERIFICATION**

  Constant `ACTION_PACKAGE_NEEDS_VERIFICATION` from class `android.content.Intent`. Sent when a package needs to be verified. The URI from `Intent.getData()` contains the package name.
- **android.intent.action.PACKAGE_REMOVED**

  Constant `ACTION_PACKAGE_REMOVED` from class `android.content.Intent`. An existing app was removed. Extra data:
  - `Intent.EXTRA_UID`: The UID of the app (`int`)

- `Intent.EXTRA_DATA_REMOVED`: Whether the data have been removed as well (Boolean)
- `Intent.EXTRA_REPLACING`: Whether an `ACTION_PACKAGE_ADDED` for the same app will follow (Boolean)

- **android.intent.action.PACKAGE_REPLACED**

  Constant `ACTION_PACKAGE_REPLACED` from class `android.content.Intent`. A new version of an existing app has been installed. The URI from `Intent.getData()` contains the package name. Extra data:

  - `Intent.EXTRA_UID` (optional): The UID of the app (`int`)

- **android.intent.action.PACKAGE_RESTARTED**

  Constant `ACTION_PACKAGE_RESTARTED` from class `android.content.Intent`. The user restarted an app. The URI from `Intent.getData()` contains the package name. Extra data:

  - `Intent.EXTRA_UID` (optional): The UID of the app (`int`)

- **android.intent.action.PACKAGE_VERIFIED**

  Constant `ACTION_PACKAGE_VERIFIED` from class `android.content.Intent`. An app has been verified. The URI from `Intent.getData()` contains the package name.

Network

- **android.intent.action.PROXY_CHANGE**

  Constant `ACTION_PROXY_CHANGE` from class `android.net.Proxy`. A proxy has changed.

- **android.net.conn.BACKGROUND_DATA_SETTING_CHANGED**

  Constant `ACTION_BACKGROUND_DATA_SETTING_CHANGED` from class `android.net.ConnectivityManager`.

  Deprecated in API level 16. Do not use it.

- **android.net.conn.CONNECTIVITY_CHANGE**

  Constant `CONNECTIVITY_ACTION` from class `android.net.ConnectivityManager`. Signals a change in the network connectivity. Contains extra data:

  - `ConnectivityManager.EXTRA_NETWORK_INFO`: Deprecated since API level 14 (android.net.`NetworkInfo`)
  - `ConnectivityManager.EXTRA_IS_FAILOVER` (optional): If this is a failover connection (Boolean)
  - `ConnectivityManager.EXTRA_NO_CONNECTIVITY` (optional): True if this is a disconnect event and there are no connected networks at all (Boolean)

- **android.net.conn.RESTRICT_BACKGROUND_CHANGED**

  Constant `ACTION_RESTRICT_BACKGROUND_CHANGED` from class `ConnectivityManager`. A change in the background metered network activity restriction has occurred.

- **android.net.nsd.STATE_CHANGED**

  Constant `ACTION_STATE_CHANGED` from class `android.net.nsd.NsdManager`. Indicate whether network service discovery is enabled or disabled. Extra data:

  – `NsdManager.EXTRA_NSD_STATE`: State information. One of:

    `NsdManager.NSD_STATE_DISABLED`

    `NsdManager.NSD_STATE_ENABLED`

- **android.net.scoring.SCORER_CHANGED**

  Constant `ACTION_SCORER_CHANGED` from class `android.net.NetworkScoreManager`. The active scorer has been changed. Extra data:

  – `NsdManager.EXTRA_NEW_SCORER`: The scorer package (`string`)

  Requires `android.Manifest.permission.SCORE_NETWORKS` permission.

- **android.net.scoring.SCORE_NETWORKS**

  Constant `ACTION_SCORE_NETWORKS` from class `android.net.NetworkScoreManager`. New network scores are being requested. Extra data:

  – `NsdManager.EXTRA_NETWORKS_TO_SCORE`: Specifies the networks to score (android.net.`NetworkKey` array)

  Requires `android.Manifest.permission.SCORE_NETWORKS` permission.

Wi-Fi

- **android.net.wifi.NETWORK_IDS_CHANGED**

  Constant `NETWORK_IDS_CHANGED_ACTION` from class `android.net.wifi.WifiManager`. Indicates a change of the IDs of the configured networks.

- **android.net.wifi.RSSI_CHANGED**

  Constant `RSSI_CHANGED_ACTION` from class `android.net.wifi.WifiManager`. The signal strength (RSSI) has changed. Extra data:

  – `WifiManager.EXTRA_NEW_RSSI`: The new RSSI in dBm (`int`)

- **android.net.wifi.SCAN_RESULTS**

  Constant `SCAN_RESULTS_AVAILABLE_ACTION` from class `android.net.wifi.WifiManager`.

  – `WifiManager.EXTRA_RESULTS_UPDATED` (optional): Whether the scan was successful (Boolean)

- **android.net.wifi.STATE_CHANGE**

  Constant `NETWORK_STATE_CHANGED_ACTION` from class `android.net.wifi.WifiManager`.

  The state of Wi-Fi connectivity has changed. Contains extra data:

  - `WifiManager.EXTRA_NETWORK_INFO`. The `android.net.NetworkInfo` object.
  - `WifiManager.EXTRA_BSSID` (optional). If the new state is `CONNECTED`, the BSSID of the access point (`string`).
  - `WifiManager.EXTRA_WIFI_INFO` (optional). If the new state is `CONNECTED`, the `android.net.wifi.WifiInfo` object.

- **android.net.wifi.WIFI_STATE_CHANGED**

  Constant `STATE_CHANGED_ACTION` from class `android.net.wifi.WifiManager`. The Wi-Fi state has been changed. Contains extra data:

  - `WifiManager.EXTRA_WIFI_STATE`: The new state. One of:

    `WifiManager.WIFI_STATE_DISABLED`

    `WifiManager.WIFI_STATE_DISABLING`

    `WifiManager.WIFI_STATE_ENABLED`

    `WifiManager.WIFI_STATE_ENABLING`

    `WifiManager.WIFI_STATE_UNKNOWN`

  - `WifiManager.EXTRA_PREVIOUS_WIFI_STATE`: The previous state. The possible values are the same as for `EXTRA_WIFI_STATE`.

- **android.net.wifi.aware.action.WIFI_AWARE_STATE_CHANGED**

  Constant `ACTION_WIFI_AWARE_STATE_CHANGED` from class `android.net.wifi.aware.WifiAwareManager`. The state of Wi-Fi Aware availability has changed.

- **android.net.wifi.p2p.CONNECTION_STATE_CHANGE**

  Constant `WIFI_P2P_CONNECTION_CHANGED_ACTION` from class `android.net.wifi.p2p.WifiP2pManager`. The state of Wi-Fi p2p connectivity has changed. Extra data:

  - `WifiP2pManager.EXTRA_WIFI_P2P_INFO`: An `android.net.wifi.p2p.WifiP2pInfo` object.
  - `WifiP2pManager.EXTRA_NETWORK_INFO`: An `android.net.NetworkInfo` object.
  - `WifiP2pManager.EXTRA_WIFI_P2P_GROUP`: An `android.net.wifi.p2p.WifiP2pGroup` object.

- **android.net.wifi.p2p.DISCOVERY_STATE_CHANGE**

  Constant `WIFI_P2P_DISCOVERY_CHANGED_ACTION` from class `android.net.wifi.p2p.WifiP2pManager`. Peer discovery has either started or stopped. Extra data:

  - `WifiP2pManager.EXTRA_DISCOVERY_STATE`: One of:

    `WifiP2pManager.WIFI_P2P_DISCOVERY_STARTED`

    `WifiP2pManager.WIFI_P2P_DISCOVERY_STOPPED`

- **android.net.wifi.p2p.PEERS_CHANGED**

  Constant `WIFI_P2P_PEERS_CHANGED_ACTION` from class `android.net.wifi.p2p.WifiP2pManager`.

  The available peer list has changed. Extra data:

  - `WifiP2pManager.EXTRA_P2P_DEVICE_LIST`: The new peer list (`android.net.wifi.p2p.WifiP2pDeviceList`)

- **android.net.wifi.p2p.STATE_CHANGED**

  Constant `WIFI_P2P_STATE_CHANGED_ACTION` from class `android.net.wifi.p2p.WifiP2pManager`. Indicates whether Wi-Fi p2p is enabled or disabled. Extra data:

  - `WifiP2pManager.EXTRA_WIFI_STATE`: One of:

    `WifiP2pManager.WIFI_P2P_STATE_DISABLED`

    `WifiP2pManager.WIFI_P2P_STATE_ENABLED`

- **android.net.wifi.p2p.THIS_DEVICE_CHANGED**

  Constant `WIFI_P2P_THIS_DEVICE_CHANGED_ACTION` from class `android.net.wifi.p2p.WifiP2pManager`. This Wi-Fi device's details have changed.

- **android.net.wifi.supplicant.CONNECTION_CHANGE**

  Constant `SUPPLICANT_CONNECTION_CHANGE_ACTION` from class `android.net.wifi.WifiManager`. A connection to the supplicant has been established or lost. Extra data:

  - `WifiP2pManager.EXTRA_SUPPLICANT_CONNECTED`: `true` if connected (Boolean).

- **android.net.wifi.supplicant.STATE_CHANGE**

  Constant `SUPPLICANT_STATE_CHANGED_ACTION` from class `android.net.wifi.WifiManager`. The state of establishing a connection to an access point has changed. Extra data:

  - WifiP2pManager.`EXTRA_NEW_STATE`: An `android.net.wifi.SupplicantState` object describing the new state.

Notifications

- **android.app.action.INTERRUPTION_FILTER_CHANGED**

  Constant `ACTION_INTERRUPTION_FILTER_CHANGED` from class `android.app.NotificationManager`. Fires when the state of `getCurrentInterruptionFilter()` changes.

- **android.app.action.NOTIFICATION_POLICY_ACCESS_GRANTED_CHANGED**

  Constant `ACTION_NOTIFICATION_POLICY_ACCESS_GRANTED_CHANGED` from class `android.app.NotificationManager`. Fires when the state of `isNotificationPolicyAccessGranted()` changes.

- **android.app.action.NOTIFICATION_POLICY_CHANGED**

  Constant `ACTION_NOTIFICATION_POLICY_CHANGED` from class `android.app.NotificationManager`. Broadcast when the state of `getNotificationPolicy()` changes.

Telephony

- **android.intent.action.CONTENT_CHANGED**

  Constant `CONTENT_CHANGED_ACTION` from class `android.provider.Telephony.Mms.Intents`. The contents of specified URIs were changed.

- **android.intent.action.DATA_SMS_RECEIVED**

  Constant `DATA_SMS_RECEIVED_ACTION` from class `android.provider.Telephony.Sms.Intents`. The device received a SMS message.
  Extra data:

  – `pdus`: An object array of byte arrays containing the PDUs.

  Requires permission `android.permission.RECEIVE_SMS`.

- **android.intent.action.NEW_OUTGOING_CALL**

  Constant `ACTION_NEW_OUTGOING_CALL` from class `android.content.Intent`. An outgoing call is being placed. Extra data:

  – `Intent.EXTRA_PHONE_NUMBER`: The original phone number being called (`string`).

    An app may replace the phone number by calling `BroadcastReceiver.setResultData(...)`. A receiver whose purpose is to prohibit phone calls should have a priority of 0 (`setPriority()` in the intent filter) to ensure it will see the final phone number to be dialed. A receiver whose purpose is to rewrite phone numbers to be called should have a positive priority. Negative priorities are reserved for the system; do not use them. Any `BroadcastReceiver` receiving this intent must not abort the broadcast. Emergency calls cannot be intercepted, and other calls cannot be modified to call

emergency numbers. If you want to redirect an outgoing call to your own service, set `resultData` to `null`, as in `BroadcastReceiver.setResultData(null)`, and then start your own app to make the call.

Requires permission `android.permission.PROCESS_OUTGOING_CALLS`.

- **android.intent.action.PHONE_STATE**

  Constant `ACTION_PHONE_STATE` from class `android.telephony.TelephonyManager`.

  The call state on the device has changed. Extra data:

  – `TelephonyManager.EXTRA_STATE`: The new state of the phone (`string`). One of:

    `TelephonyManager.EXTRA_STATE_IDLE`: The phone is idle.

    `TelephonyManager.EXTRA_STATE_RINGING`: A new call arrived and is ringing or waiting.

    `TelephonyManager.EXTRA_STATE_OFFHOOK`: At least one call exists that is dialing, active, or on hold, and no calls are ringing or waiting.

  – `TelephonyManager.EXTRA_INCOMING_NUMBER` (optional): If the state is `RINGING`, this field contains the number (`string`).

  Requires permission `android.permission.READ_PHONE_STATE`.

- **android.provider.Telephony.SIM_FULL**

  Constant `SIM_FULL_ACTION` from class `android.provider.Telephony`. The SIM storage for SMS messages is full.

- **android.provider.Telephony.SMS_CB_RECEIVED**

  Constant `SMS_CB_RECEIVED_ACTION` from class `android.provider.Telephony`. A new Cell Broadcast message has been received. Extra data:

  – `message`: An `android.telephony.SmsCbMessage` object containing the broadcast message.

- **android.provider.Telephony.SMS_EMERGENCY_CB_RECEIVED**

  Constant `SMS_EMERGENCY_CB_RECEIVED` from class `android.provider.Telephony`. A new Emergency Broadcast message has been received. Extra data:

  – `message`: A `android.telephony.SmsCbMessage` object containing the broadcast message, including ETWS or CMAS warning notification info if present.

- **android.provider.Telephony.SMS_DELIVER**

  Constant `SMS_DELIVER_ACTION` from class `android.provider.Telephony`. An SMS has been received. This broadcast goes only to the default app. Extra data:

  – `pdus`: An `Object[]` of `byte[]`s containing the PDUs

  Requires permission `android.Manifest.permission.BROADCAST_SMS`.

- **android.provider.Telephony.SMS_RECEIVED**

  Constant `SMS_RECEIVED_ACTION` from class `android.provider.Telephony`. An SMS has been received. Extra data:

  - `pdus`: an `Object[]` of `byte[]`s containing the PDUs

- **android.provider.Telephony.SMS_REJECTED**

  Constant `SMS_REJECTED_ACTION` from class `android.provider.Telephony`. An SMS has been rejected by the telephony framework. Extra data:

  - `result`: The reason for the rejection. One of:

    `Telephony.RESULT_SMS_GENERIC_ERROR`: Generic error

    `Telephony.RESULT_SMS_OUT_OF_MEMORY`: Device out of memory

    `Telephony.RESULT_SMS_UNSUPPORTED`: Format or encoding not supported

    `Telephony.RESULT_SMS_DUPLICATED`: Message duplicated

- **android.provider.Telephony.SMS_SERVICE_CATEGORY_PROGRAM_DATA_RECEIVED**

  Constant `SMS_SERVICE_CATEGORY_PROGRAM_DATA_RECEIVED_ACTION` from class `android.provider.Telephony`.

  A new CDMA SMS has been received containing *Service Category Program Data*. Extra data:

  - `operations`: An array of `android.telephony.cdma.CdmaSmsCbProgramData` objects containing the service category operations (add/delete/clear) to perform.

- **android.provider.Telephony.WAP_PUSH_DELIVER**

  Constant `WAP_PUSH_DELIVER_ACTION` from class `android.provider.Telephony`.

  A new WAP PUSH message has been received. This broadcast will only be delivered to the default SMS app. Extra data:

  - `transactionId`: The WAP transaction ID (`int`)
  - `pduType`: The WAP PDU type (`int`)
  - `header`: The message header (`byte[]`)
  - `data`: The data payload (`byte[]`)
  - `contentTypeParameters`: Parameters associated with the content type (`Map<String,String>`)

  Requires `android.Manifest.permission.BROADCAST_WAP_PUSH` permission.

- **android.provider.Telephony.WAP_PUSH_RECEIVED**

  Constant `WAP_PUSH_RECEIVED_ACTION` from class `android.provider.Telephony` .

A new WAP PUSH message has been received by the device. Extra data:

- `transactionId`: The WAP transaction ID (`int`)
- `pduType`: The WAP PDU type (`int`)
- `header`: The message header (`byte[]`)
- `data`: The data payload (`byte[]`)
- `contentTypeParameters`: Parameters associated with the content type (`Map<String,String>`)

- **android.provider.action.DEFAULT_SMS_PACKAGE_CHANGED**

  Constant `ACTION_DEFAULT_SMS_PACKAGE_CHANGED` from class `android.provider.Telephony`. With the default SMS app, both the new and the previous package receive this broadcast. Extra data:

  - `Telephony.EXTRA_IS_DEFAULT_SMS_APP`: True if this is the new default SMS app (Boolean)

- **android.provider.action.EXTERNAL_PROVIDER_CHANGE**

  Constant `ACTION_EXTERNAL_PROVIDER_CHANGE` from class `android.provider.Telephony`.

  A change is made to the `SmsProvider` or `MmsProvider` by a process other than the default SMS application. The URI used for that can be read by `Intent.getData()`.

- **android.telephony.action.DEFAULT_SMS_SUBSCRIPTION_CHANGED**

  Constant `ACTION_DEFAULT_SMS_SUBSCRIPTION_CHANGED` from class `android.telephony.SubscriptionManager`. The default SMS subscription has changed. Extra data:

  - `SubscriptionManager.EXTRA_SUBSCRIPTION_INDEX`: Indicates which subscription has changed (`int`).

- **android.telephony.action.DEFAULT_SUBSCRIPTION_CHANGED**

  Constant `ACTION_DEFAULT_SUBSCRIPTION_CHANGED` from class `android.telephony.SubscriptionManager`. The default subscription has changed. Extra data:

  - `SubscriptionManager.EXTRA_SUBSCRIPTION_INDEX`: Indicates which subscription has changed (`int`).

TV

- **android.media.tv.action.INITIALIZE_PROGRAMS**

  Constant `ACTION_INITIALIZE_PROGRAMS` from class `android.media.tv.TvContract`. Sent to the target TV input after it is first installed to notify the input to initialize its channels and programs to the system content provider.

- **android.media.tv.action.PREVIEW_PROGRAM_ADDED_TO_WATCH_NEXT**

  Constant `ACTION_PREVIEW_PROGRAM_ADDED_TO_WATCH_NEXT` from class `android.media.tv.TvContract`. Sent by the system to tell the target TV that one of its existing preview programs is added to the "watch next" programs table. Extra data:

  - `TvContract.EXTRA_PREVIEW_PROGRAM_ID`: The ID of the existing preview program (`long`)
  - `TvContract.EXTRA_WATCH_NEXT_PROGRAM_ID`: The ID of the new "watch next" program (`long`).

- **android.media.tv.action.PREVIEW_PROGRAM_BROWSABLE_DISABLED**

  Constant `ACTION_PREVIEW_PROGRAM_BROWSABLE_DISABLED` from class `android.media.tv.TvContract`. Sent by the system to tell the target TV that one of its preview program's browsable state is disabled. This means it will no longer be shown to users. Extra data:

  - `TvContract.EXTRA_PREVIEW_PROGRAM_ID`: The ID of the preview program (`long`)

- **android.media.tv.action.WATCH_NEXT_PROGRAM_BROWSABLE_DISABLED**

  Constant `ACTION_WATCH_NEXT_PROGRAM_BROWSABLE_DISABLED` from class `android.media.tv.TvContract`. Sent by the system to tell the target TV that one of its "watch next" program's browsable state is disabled, that is it will no longer be shown to users. Extra data:

  - `TvContract.EXTRA_WATCH_NEXT_PROGRAM_ID`: The disabled program ID (`long`)

Audio

- **android.intent.action.HEADSET_PLUG**

  Constant `ACTION_HEADSET_PLUG` from class `android.content.Intent`. Same as `android.intent.action.HEADSET_PLUG` for class `AudioManager`.

- **android.intent.action.HEADSET_PLUG**

  Constant `ACTION_HEADSET_PLUG` from class `android.media.AudioManager`. A headset was plugged or unplugged. Extra data:

  - `state`: 0 for unplugged, 1 for plugged (`int`)
  - `name`: Human-readable name (`string`)
  - `microphone`: 1 if it has a microphone, 0 if not (`int`)

- **android.media.ACTION_SCO_AUDIO_STATE_UPDATED**

  Constant `ACTION_SCO_AUDIO_STATE_UPDATED` from class `android.media.AudioManager`.

The Bluetooth SCO audio connection state has been updated. Extra data:

- `AudioManager.EXTRA_SCO_AUDIO_STATE`: The new SCO audio state. One of:

  `AudioManager.SCO_AUDIO_STATE_DISCONNECTED`

  `AudioManager.SCO_AUDIO_STATE_CONNECTING`

  `AudioManager.SCO_AUDIO_STATE_CONNECTED`

- `AudioManager.EXTRA_SCO_AUDIO_PREVIOUS_STATE`: The new SCO audio state. The possible values are the same as for `EXTRA_SCO_AUDIO_STATE`.

- **android.media.AUDIO_BECOMING_NOISY**

  Constant `ACTION_AUDIO_BECOMING_NOISY` from class `android.media.AudioManager`. A hint for applications that audio is about to become noisy.

- **android.media.RINGER_MODE_CHANGED**

  Constant `RINGER_MODE_CHANGED_ACTION` from class `android.media.AudioManager`. Indicates that the ringer mode has changed. Extra data:

  - `AudioManager.EXTRA_RINGER_MODE`: One of:

    `AudioManager.RINGER_MODE_NORMAL`

    `AudioManager.RINGER_MODE_SILENT`

    `AudioManager.RINGER_MODE_VIBRATE`

- **android.media.SCO_AUDIO_STATE_CHANGED**

  Deprecated in API level 14. Use `SCO_AUDIO_STATE_UPDATED` instead.

- **android.media.VIBRATE_SETTING_CHANGED**

  Deprecated in API level 16. Use `RINGER_MODE_CHANGED_ACTION` instead.

- **android.media.action.CLOSE_AUDIO_EFFECT_CONTROL_SESSION**

  Constant `ACTION_CLOSE_AUDIO_EFFECT_CONTROL_SESSION` from class `android.media.audiofx.AudioEffect`. Signals that an audio session is closed and that effects should not be applied anymore. Contains extra data:

  - `AudioEffect.EXTRA_PACKAGE_NAME`: Calling package (`string`)
  - `AudioEffect.EXTRA_AUDIO_SESSION`: The session ID (`int`)

- **android.media.action.HDMI_AUDIO_PLUG**

  Constant `ACTION_HDMI_AUDIO_PLUG` from class `android.media.AudioManager`. An HDMI cable was plugged or unplugged. Extra data:

  - `AudioManager.EXTRA_AUDIO_PLUG_STATE`: 1 = plugged-in, 0 = unplugged (`int`).
  - `AudioManager.EXTRA_MAX_CHANNEL_COUNT`: Maximum number of channels (`int`).

  - `AudioManager.EXTRA_ENCODINGS`: The audio encodings supported by the connected HDMI device (`int` array). The values are as those fields in `android.media.AudioFormat` starting with `ENCODING_`.

- **android.media.action.OPEN_AUDIO_EFFECT_CONTROL_SESSION**

  Constant `ACTION_OPEN_AUDIO_EFFECT_CONTROL_SESSION` from class `android.media.audiofx.AudioEffect`. Signals that an audio session is opened and that effects should be applied. Contains extra data:

  - `AudioEffect.EXTRA_PACKAGE_NAME`: The calling package (`string`)
  - `AudioEffect.EXTRA_AUDIO_SESSION`: The session ID (`int`)

Booting and Shutdown

- **android.intent.action.ACTION_SHUTDOWN**

  Constant `ACTION_SHUTDOWN` from class `android.content.Intent`. Device about to be shut down. May contain extra data:

  - `Intent.EXTRA_SHUTDOWN_USERSPACE_ONLY`: A Boolean; `true` if this is only a shutdown of user processes. The default is `false`.

- **android.intent.action.BOOT_COMPLETED**

  Constant `ACTION_BOOT_COMPLETED` from class `android.content.Intent`. The device has finished booting. Requires permission `android.permission.RECEIVE_BOOT_COMPLETED`.

- **android.intent.action.LOCKED_BOOT_COMPLETED**

  Constant `ACTION_LOCKED_BOOT_COMPLETED` from class `android.content.Intent`. The device has been booted but is still locked. If you react to this broadcast, only the device protected storage can be accessed if you need a data storage. Requires `android.permission.RECEIVE_BOOT_COMPLETED` permission.

- **android.intent.action.REBOOT**

  Constant `ACTION_REBOOT` from class `android.content.Intent`. Have the device do a reboot. Only system code; do not use.

Security

- **android.security.STORAGE_CHANGED**

  Constant `ACTION_STORAGE_CHANGED` from class `android.security.KeyChain`. Deprecated in API level 26. Instead use the more fine-grained events `ACTION_KEYCHAIN_CHANGED`, `ACTION_TRUST_STORE_CHANGED`, or `ACTION_KEY_ACCESS_CHANGED`.

- **android.security.action.KEYCHAIN_CHANGED**

  Constant `ACTION_KEYCHAIN_CHANGED` from class `android.security.KeyChain`. The contents of the keychain have changed.

- **android.security.action.KEY_ACCESS_CHANGED**

  Constant `ACTION_KEY_ACCESS_CHANGED` from class `android.security.KeyChain`. The access permissions for a private key have changed.

- **android.security.action.TRUST_STORE_CHANGED**

  Constant `ACTION_TRUST_STORE_CHANGED` from class `android.security.KeyChain`. The contents of the trusted certificate store has changed. Fires when a pre-installed CA is disabled or re-enabled or a CA is added or removed from the trust store.

Various

- **android.app.action.APPLICATION_DELEGATION_SCOPES_CHANGED**

  Constant `ACTION_APPLICATION_DELEGATION_SCOPES_CHANGED` from class `android.app.admin.DevicePolicyManager`. A broadcast sent after application delegation scopes are changed. The new scopes are listed in a `string` array from extra data with key `DevicePolicyManager.EXTRA_DELEGATION_SCOPES`.

- **android.app.action.LOCK_TASK_ENTERING**

  Constant `ACTION_LOCK_TASK_ENTERING` from class `android.app.admin.DeviceAdminReceiver`. The device is entering lock task mode. Extra data:

  - `DeviceAdminReceiver.EXTRA_LOCK_TASK_PACKAGE`: The name of the package using lock task mode (`string`).

- **android.app.action.LOCK_TASK_EXITING**

  Constant `ACTION_LOCK_TASK_EXITING` from class `android.app.admin.DeviceAdminReceiver`. The device is exiting from lock task mode.

- **android.app.action.NEXT_ALARM_CLOCK_CHANGED**

  Constant `ACTION_NEXT_ALARM_CLOCK_CHANGED` from class `android.app.AlarmManager`. Sent after the value returned by `getNextAlarmClock()` has changed.

- **android.app.action.SYSTEM_UPDATE_POLICY_CHANGED**

  Constant `ACTION_SYSTEM_UPDATE_POLICY_CHANGED` from class `android.app.admin.DevicePolicyManager`. A new local system update policy has been set by the device owner.

- **android.content.pm.action.SESSION_COMMITTED**

  Constant `ACTION_SESSION_COMMITTED` from class `android.content.pm.PackageInstaller`. Explicit broadcast sent to the last known default launcher when a session for a new install is committed. Contains extra data:

  - `EXTRA_SESSION`: The session (`android.content.pm.PackageInstaller.SessionInfo`).
  - `android.content.Intent.EXTRA_USER`: The user (`android.os.UserHandle`)

- **android.hardware.action.NEW_PICTURE**

  Constant `ACTION_NEW_PICTURE` from class `android.hardware.Camera`. Deprecated in API level 21. Brought back in API level 26. A new picture is taken. Use `Intent.getData()` to get the URI of the picture.

- **android.hardware.action.NEW_VIDEO**

  Constant `ACTION_NEW_VIDEO` from class `android.hardware.Camera`. Deprecated in API level 21. Brought back in API level 26. A new video is recorded. Use `Intent.getData()` to get the URI of the video.

- **android.hardware.hdmi.action.OSD_MESSAGE**

  Sent when the `HdmiControlManager` service has a message to display on screen for events that need user's attention such as ARC status change. Extra data:

  - `android.hardware.hdmi.extra.MESSAGE_ID`: The ID of the message to display on-screen.

  Requires `android.Manifest.permission.HDMI_CEC` permission.

- **android.hardware.input.action.QUERY_KEYBOARD_LAYOUTS**

  Constant `ACTION_QUERY_KEYBOARD_LAYOUTS` from class `android.hardware.input.InputManager`.

  The input manager service locates available keyboard layouts by querying broadcast receivers that are registered for this action. For details, see the API documentation of `android.hardware.input.InputManager`.

- **android.intent.action.AIRPLANE_MODE**

  Constant `ACTION_AIRPLANE_MODE` from class `android.content.Intent`. The user-toggled airplane mode. Contains extra data:

  - `state`: A Boolean; `true` if airplane mode is on.

- **android.intent.action.APPLICATION_RESTRICTIONS_CHANGED**

  Constant `ACTION_APPLICATION_RESTRICTIONS_CHANGED` from class `android.content.Intent`. Sent after application restrictions are changed.

- **android.intent.action.CAMERA_BUTTON**

  Constant `ACTION_CAMERA_BUTTON` from class `android.content.Intent`. The camera button was pressed. Extra data:

  - `Intent.EXTRA_KEY_EVENT`: The associated `android.view.Key-Event`.

- **android.intent.action.CLOSE_SYSTEM_DIALOGS**

  Constant `ACTION_CLOSE_SYSTEM_DIALOGS` from class `android.content.Intent`. A user action should request a temporary system dialog to dismiss.

- **android.intent.action.CONFIGURATION_CHANGED**

  Constant `ACTION_CONFIGURATION_CHANGED` from class `android.content.Intent`. The current device configuration (orientation, locale, etc.), represented by class `android.content.res.Configuration`, has changed.

- **android.intent.action.DATE_CHANGED**

  Constant `ACTION_DATE_CHANGED` from class `android.content.Intent`. The date has changed.

- **android.intent.action.DEVICE_STORAGE_LOW**

  Constant `ACTION_DEVICE_STORAGE_LOW` from class `android.content.Intent`. Defunct in API level 26 (Android 8.0).

- **android.intent.action.DEVICE_STORAGE_OK**

  Constant `ACTION_DEVICE_STORAGE_OK` from class `android.content.Intent`. Defunct in API level 26 (Android 8.0).

- **android.intent.action.DOCK_EVENT**

  Constant `ACTION_DOCK_EVENT` from class `android.content.Intent`.

  A change in the physical docking state occurred. Extra data:

  - `Intent.EXTRA_DOCK_STATE`: The dock state. Possible values are as follows:

    `Intent.EXTRA_DOCK_STATE_UNDOCKED`: Undocked

    `Intent.EXTRA_DOCK_STATE_DESK`: Desk dock

    `Intent.EXTRA_DOCK_STATE_CAR`: Car dock

    `Intent.EXTRA_DOCK_STATE_LE_DESK`: Low performance (analog) dock

    `Intent.EXTRA_DOCK_STATE_HE_DESK`: High performance (digital) dock.

- **android.intent.action.DOWNLOAD_COMPLETE**

  Constant `ACTION_DOWNLOAD_COMPLETE` from class `android.app.DownloadManager`. A download completed.

- **android.intent.action.DOWNLOAD_NOTIFICATION_CLICKED**

  Constant `ACTION_DOWNLOAD_NOTIFICATION_CLICKED` from class `android.app.DownloadManager`. The user clicks a running download.

- **android.intent.action.DREAMING_STARTED**

  Undocumented. See `android.service.dreams.DreamService` for a service approach to dreaming (custom interactive screensaver when the device is idle).

- **android.intent.action.DREAMING_STOPPED**

  Undocumented. See `android.service.dreams.DreamService` for a service approach to dreaming (custom interactive screensaver when the device is idle).

- **android.intent.action.DROPBOX_ENTRY_ADDED**

  Constant `ACTION_DROPBOX_ENTRY_ADDED` from class `android.os.DropBoxManager`.

  A new entry is added in the dropbox. Requires `android.permission.READ_LOGS` permission.

- **android.intent.action.EXTERNAL_APPLICATIONS_AVAILABLE**

  Constant `ACTION_EXTERNAL_APPLICATIONS_AVAILABLE` from class `android.content.Intent`.

  Previously unavailable apps are now available, for example because a medium has been attached. Extra data:

  - `Intent.EXTRA_CHANGED_PACKAGE_LIST`: List of new apps available (`string` array)
  - `Intent.EXTRA_CHANGED_UID_LIST`: List of corresponding UIDs (`int` array)

- **android.intent.action.EXTERNAL_APPLICATIONS_UNAVAILABLE**

  Constant `ACTION_EXTERNAL_APPLICATIONS_UNAVAILABLE` from class `android.content.Intent`.

  Previously available apps are now unavailable, for example because a medium has been removed. Extra data:

  - `Intent.EXTRA_CHANGED_PACKAGE_LIST`: List of new apps available (`string` array)
  - `Intent.EXTRA_CHANGED_UID_LIST`: List of corresponding UIDs (`int` array)

- **android.intent.action.FETCH_VOICEMAIL**

  Constant `ACTION_FETCH_VOICEMAIL` from class `android.provider.VoicemailContract`.

Use this to request a voicemail source to fetch voicemail content from the remote server. The voicemail to fetch is specified by the data URI of the intent.

- **android.intent.action.GTALK_CONNECTED**

  Constant `ACTION_GTALK_SERVICE_CONNECTED` from class `android.content.Intent`.

  A GTalk connection has been established.

- **android.intent.action.GTALK_DISCONNECTED**

  Constant `ACTION_GTALK_SERVICE_DISCONNECTED` from class `android.content.Intent`.

  A GTalk connection has been closed.

- **android.intent.action.INPUT_METHOD_CHANGED**

  Constant `ACTION_INPUT_METHOD_CHANGED` from class `android.content.Intent`.

  An input method has been changed.

- **android.intent.action.LOCALE_CHANGED**

  Constant `LOCALE_CHANGED` from class `android.content.Intent`.

  The device's locale has been changed.

- **android.intent.action.MANAGE_PACKAGE_STORAGE**

  Constant `ACTION_MANAGE_PACKAGE_STORAGE` from class `android.content.Intent`.

  The low memory condition notification acknowledged by user and package management should be started.

  – `intent.EXTRA_USER`: The corresponding user (`android.os.UserHandle`).

- **android.intent.action.NEW_VOICEMAIL**

  Constant `ACTION_NEW_VOICEMAIL` from class `android.provider.VoicemailContract`.

  A new voicemail record was inserted.

- **android.intent.action.PROVIDER_CHANGED**

  Constant `ACTION_PROVIDER_CHANGED` from class `android.content.Intent`.

  A content provider signals a data change. The URI from `Intent.getData()` contains information about the change, and besides in the extra data we have the following:

  – `count`: The number of items in the data set (`int`).

- **android.intent.action.SCREEN_OFF**

  Constant `ACTION_SCREEN_OFF` from class `android.content.Intent`. The device goes to sleep and becomes non-interactive.

- **android.intent.action.SCREEN_ON**

  Constant `ACTION_SCREEN_ON` from class `android.content.Intent`. The device becomes interactive again after being non-interactive.

- **android.intent.action.TIMEZONE_CHANGED**

  Constant `ACTION_TIMEZONE_CHANGED` from class `android.content.Intent`. The time zone has changed. Extra data:

  – `time-zone`: The `java.util.TimeZone.getID()` value (`string`).

- **android.intent.action.TIME_SET**

  Constant `ACTION_TIME_CHANGED` from class `android.content.Intent`. The time was set.

- **android.intent.action.TIME_TICK**

  Constant `ACTION_TIME_TICK` from class `android.content.Intent`. The time has changed; it is sent every minute.

- **android.intent.action.WALLPAPER_CHANGED**

  Constant `ACTION_WALLPAPER_CHANGED` from class `android.content.Intent`.

  Deprecated in API level 16. Do not use.

- **android.nfc.action.ADAPTER_STATE_CHANGED**

  Constant `ACTION_ADAPTER_STATE_CHANGED` from class `android.nfc.NfcAdapter`.

  The state of the local NFC adapter has been changed. Extra data:

  – `NfcAdapter.EXTRA_ADAPTER_STATE`: The new state. One of:

  `NfcAdapter.STATE_OFF`

  `NfcAdapter.STATE_TURNING_ON`

  `NfcAdapter.STATE_ON`

  `NfcAdapter.STATE_TURNING_OFF`

- **android.provider.action.SYNC_VOICEMAIL**

  Constant `ACTION_SYNC_VOICEMAIL` from class `android.provider.VoicemailContract`.

  Used to request all voicemail sources to perform a sync with the remote server.

- **android.speech.tts.TTS_QUEUE_PROCESSING_COMPLETED**

  Constant `ACTION_TTS_QUEUE_PROCESSING_COMPLETED` from class `android.speech.tts.TextToSpeech`.

  The `TextToSpeech` synthesizer has completed processing of all the text in the speech queue. This does not mean any sound has been generated yet.

- **android.speech.tts.engine.TTS_DATA_INSTALLED**

  Constant `ACTION_TTS_DATA_INSTALLED` from class `android.speech.tts.TextToSpeech.Engine`.

  Signals the change in the list of available languages or their features.