

public class

**Intent**extends [Object](#)implements [Parcelable](#)[Cloneable](#)Summary: [Nested Classes](#) | [Constants](#) | [Inherited Constants](#) | [Fields](#) |[Ctors](#) | [Methods](#) | [Inherited Methods](#) | [Expand All](#)Added in **API level 1**[java.lang.Object](#)↳ [android.content.Intent](#)

► Known Direct Subclasses

[LabeledIntent](#)

## Class Overview

An intent is an abstract description of an operation to be performed. It can be used with [startActivity\(\)](#) ([reference/android/content/Context.html#startActivity\(android.content.Intent\)](#)) to launch an [Activity](#) ([reference/android/app/Activity.html](#)), [broadcastIntent\(\)](#) ([reference/android/content/Context.html#sendBroadcast\(android.content.Intent\)](#)) to send it to any interested [BroadcastReceiver](#) ([reference/android/content/BroadcastReceiver.html](#)) components, and [startService\(Intent\)](#) ([reference/android/content/Context.html#startService\(android.content.Intent\)](#)) or [bindService\(Intent, ServiceConnection, int\)](#) ([reference/android/content/Context.html#bindService\(android.content.Intent, android.content.ServiceConnection, int\)](#)) to communicate with a background [Service](#) ([reference/android/app/Service.html](#)).

An Intent provides a facility for performing late runtime binding between the code in different applications. Its most significant use is in the launching of activities, where it can be thought of as the glue between activities. It is basically a passive data structure holding an abstract description of an action to be performed.

### Developer Guides

For information about how to create and resolve intents, read the [Intents and Intent Filters](#) ([/guide/topics/intents/intents-filters.html](#)) developer guide.

### Intent Structure

The primary pieces of information in an intent are:

- **action** -- The general action to be performed, such as [ACTION\\_VIEW](#) ([reference/android/content/Intent.html#ACTION\\_VIEW](#)), [ACTION\\_EDIT](#) ([reference/android/content/Intent.html#ACTION\\_EDIT](#)), [ACTION\\_MAIN](#) ([reference/android/content/Intent.html#ACTION\\_MAIN](#)), etc.
- **data** -- The data to operate on, such as a person record in the contacts database, expressed as a [Uri](#) ([reference/android/net/Uri.html](#)).

Some examples of action/data pairs are:

- [ACTION\\_VIEW](#) ([reference/android/content/Intent.html#ACTION\\_VIEW](#)) *content://contacts/people/1* -- Display information about the person whose identifier is "1".
- [ACTION\\_DIAL](#) ([reference/android/content/Intent.html#ACTION\\_DIAL](#)) *content://contacts/people/1* -- Display the phone dialer with the person filled in.
- [ACTION\\_VIEW](#) ([reference/android/content/Intent.html#ACTION\\_VIEW](#)) *tel:123* -- Display the phone dialer with the given number filled in. Note how the VIEW action does what what is considered the most reasonable thing for a particular URI.

- **ACTION\_DIAL** ([/reference/android/content/Intent.html#ACTION\\_DIAL](/reference/android/content/Intent.html#ACTION_DIAL)) *tel:123* -- Display the phone dialer with the given number filled in.
- **ACTION\_EDIT** ([/reference/android/content/Intent.html#ACTION\\_EDIT](/reference/android/content/Intent.html#ACTION_EDIT)) *content://contacts/people/1* -- Edit information about the person whose identifier is "1".
- **ACTION\_VIEW** ([/reference/android/content/Intent.html#ACTION\\_VIEW](/reference/android/content/Intent.html#ACTION_VIEW)) *content://contacts/people/* -- Display a list of people, which the user can browse through. This example is a typical top-level entry into the Contacts application, showing you the list of people. Selecting a particular person to view would result in a new intent { **ACTION\_VIEW** ([/reference/android/content/Intent.html#ACTION\\_VIEW](/reference/android/content/Intent.html#ACTION_VIEW)) *content://contacts/N* } being used to start an activity to display that person.

In addition to these primary attributes, there are a number of secondary attributes that you can also include with an intent:

- **category** -- Gives additional information about the action to execute. For example, **CATEGORY\_LAUNCHER** ([/reference/android/content/Intent.html#CATEGORY\\_LAUNCHER](/reference/android/content/Intent.html#CATEGORY_LAUNCHER)) means it should appear in the Launcher as a top-level application, while **CATEGORY\_ALTERNATIVE** ([/reference/android/content/Intent.html#CATEGORY\\_ALTERNATIVE](/reference/android/content/Intent.html#CATEGORY_ALTERNATIVE)) means it should be included in a list of alternative actions the user can perform on a piece of data.
- **type** -- Specifies an explicit type (a MIME type) of the intent data. Normally the type is inferred from the data itself. By setting this attribute, you disable that evaluation and force an explicit type.
- **component** -- Specifies an explicit name of a component class to use for the intent. Normally this is determined by looking at the other information in the intent (the action, data/type, and categories) and matching that with a component that can handle it. If this attribute is set then none of the evaluation is performed, and this component is used exactly as is. By specifying this attribute, all of the other Intent attributes become optional.
- **extras** -- This is a **Bundle** (</reference/android/os/Bundle.html>) of any additional information. This can be used to provide extended information to the component. For example, if we have an action to send an e-mail message, we could also include extra pieces of data here to supply a subject, body, etc.

Here are some examples of other operations you can specify as intents using these additional parameters:

- **ACTION\_MAIN** ([/reference/android/content/Intent.html#ACTION\\_MAIN](/reference/android/content/Intent.html#ACTION_MAIN)) with category **CATEGORY\_HOME** ([/reference/android/content/Intent.html#CATEGORY\\_HOME](/reference/android/content/Intent.html#CATEGORY_HOME)) -- Launch the home screen.
- **ACTION\_GET\_CONTENT** ([/reference/android/content/Intent.html#ACTION\\_GET\\_CONTENT](/reference/android/content/Intent.html#ACTION_GET_CONTENT)) with MIME type *vnd.android.cursor.item/phone* ([/reference/android/provider/Contacts.Phones.html#CONTENT\\_URI](/reference/android/provider/Contacts.Phones.html#CONTENT_URI)) -- Display the list of people's phone numbers, allowing the user to browse through them and pick one and return it to the parent activity.
- **ACTION\_GET\_CONTENT** ([/reference/android/content/Intent.html#ACTION\\_GET\\_CONTENT](/reference/android/content/Intent.html#ACTION_GET_CONTENT)) with MIME type */\** and category **CATEGORY\_OPENABLE** ([/reference/android/content/Intent.html#CATEGORY\\_OPENABLE](/reference/android/content/Intent.html#CATEGORY_OPENABLE)) -- Display all pickers for data that can be opened with **ContentResolver.openInputStream()** ([/reference/android/content/ContentResolver.html#openInputStream\(android.net.Uri\)](/reference/android/content/ContentResolver.html#openInputStream(android.net.Uri))), allowing the user to pick one of them and then some data inside of it and returning the resulting URI to the caller. This can be used, for example, in an e-mail application to allow the user to pick some data to include as an attachment.

There are a variety of standard Intent action and category constants defined in the Intent class, but applications can also define their own. These strings use java style scoping, to ensure they are unique -- for example, the standard **ACTION\_VIEW** ([/reference/android/content/Intent.html#ACTION\\_VIEW](/reference/android/content/Intent.html#ACTION_VIEW)) is called "android.intent.action.VIEW".

Put together, the set of actions, data types, categories, and extra data defines a language for the system allowing for the expression of phrases such as "call john smith's cell". As applications are added to the system, they can extend this language by adding new actions, types, and categories, or they can modify the behavior of existing phrases by supplying their own activities that handle them.

## Intent Resolution

There are two primary forms of intents you will use.

- **Explicit Intents** have specified a component (via `setComponent(ComponentName)` ([`/reference/android/content/Intent.html#setComponent\(android.content.ComponentName\)`](/reference/android/content/Intent.html#setComponent(android.content.ComponentName))) or `setClass(Context, Class)` ([`/reference/android/content/Intent.html#setClass\(android.content.Context, java.lang.Class<?>\(\)\)`](/reference/android/content/Intent.html#setClass(android.content.Context, java.lang.Class<?>()))), which provides the exact class to be run. Often these will not include any other information, simply being a way for an application to launch various internal activities it has as the user interacts with the application.
- **Implicit Intents** have not specified a component; instead, they must include enough information for the system to determine which of the available components is best to run for that intent.

When using implicit intents, given such an arbitrary intent we need to know what to do with it. This is handled by the process of *Intent resolution*, which maps an Intent to an [`Activity`](/reference/android/app/Activity.html) ([`/reference/android/app/Activity.html`](/reference/android/app/Activity.html)), [`BroadcastReceiver`](/reference/android/content/BroadcastReceiver.html) ([`/reference/android/content/BroadcastReceiver.html`](/reference/android/content/BroadcastReceiver.html)), or [`Service`](/reference/android/app/Service.html) ([`/reference/android/app/Service.html`](/reference/android/app/Service.html)) (or sometimes two or more activities/receivers) that can handle it.

The intent resolution mechanism basically revolves around matching an Intent against all of the `<intent-filter>` descriptions in the installed application packages. (Plus, in the case of broadcasts, any [`BroadcastReceiver`](/reference/android/content/BroadcastReceiver.html) ([`/reference/android/content/BroadcastReceiver.html`](/reference/android/content/BroadcastReceiver.html)) objects explicitly registered with `registerReceiver(BroadcastReceiver, IntentFilter)` ([`/Context.html#registerReceiver\(android.content.BroadcastReceiver, android.content.IntentFilter\)`](/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter)).) More details on this can be found in the documentation on the [`IntentFilter`](/reference/android/content/IntentFilter.html) ([`/reference/android/content/IntentFilter.html`](/reference/android/content/IntentFilter.html)) class.

There are three pieces of information in the Intent that are used for resolution: the action, type, and category. Using this information, a query is done on the [`PackageManager`](/reference/android/content/pm/PackageManager.html) ([`/reference/android/content/pm/PackageManager.html`](/reference/android/content/pm/PackageManager.html)) for a component that can handle the intent. The appropriate component is determined based on the intent information supplied in the `AndroidManifest.xml` file as follows:

- The **action**, if given, must be listed by the component as one it handles.
- The **type** is retrieved from the Intent's data, if not already supplied in the Intent. Like the action, if a type is included in the intent (either explicitly or implicitly in its data), then this must be listed by the component as one it handles.
- For data that is not a content: URI and where no explicit type is included in the Intent, instead the **scheme** of the intent data (such as `http:` or `mailto:`) is considered. Again like the action, if we are matching a scheme it must be listed by the component as one it can handle.
- The **categories**, if supplied, must *all* be listed by the activity as categories it handles. That is, if you include the categories [`CATEGORY\_LAUNCHER`](/reference/android/content/Intent.html#CATEGORY_LAUNCHER) ([`/reference/android/content/Intent.html#CATEGORY\_LAUNCHER`](/reference/android/content/Intent.html#CATEGORY_LAUNCHER)) and [`CATEGORY\_ALTERNATIVE`](/reference/android/content/Intent.html#CATEGORY_ALTERNATIVE) ([`/reference/android/content/Intent.html#CATEGORY\_ALTERNATIVE`](/reference/android/content/Intent.html#CATEGORY_ALTERNATIVE)), then you will only resolve to components with an intent that lists *both* of those categories. Activities will very often need to support the [`CATEGORY\_DEFAULT`](/reference/android/content/Intent.html#CATEGORY_DEFAULT) ([`/reference/android/content/Intent.html#CATEGORY\_DEFAULT`](/reference/android/content/Intent.html#CATEGORY_DEFAULT)) so that they can be found by `Context.startActivity()` ([`/Context.html#startActivity\(android.content.Intent\)`](/reference/android/content/Context.html#startActivity(android.content.Intent))).

For example, consider the Note Pad sample application that allows user to browse through a list of notes data and view details about individual items. Text in *italics* indicate places where you would replace a name with one specific to your own package.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.notepad">
    <application android:icon="@drawable/app_notes"
        android:label="@string/app_name">

        <provider class=".NotePadProvider"
            android:authorities="com.google.provider.NotePad" /

        <activity class=".NotesList" android:label="@string/title_n
            <intent-filter>
                <action android:name="android.intent.action.MAIN" /
                <category android:name="android.intent.category.LAU
            </intent-filter>
            <intent-filter>
                <action android:name="android.intent.action.VIEW" /
                <action android:name="android.intent.action.EDIT" /
                <action android:name="android.intent.action.PICK" /
                <category android:name="android.intent.category.DEF
                <data android:mimeType="vnd.android.cursor.dir/vnd.
            </intent-filter>
            <intent-filter>
                <action android:name="android.intent.action.GET_CON
                <category android:name="android.intent.category.DEF
                <data android:mimeType="vnd.android.cursor.item/vnd
            </intent-filter>
        </activity>

        <activity class=".NoteEditor" android:label="@string/title_
            <intent-filter android:label="@string/resolve_edit">
                <action android:name="android.intent.action.VIEW" /
                <action android:name="android.intent.action.EDIT" /
                <category android:name="android.intent.category.DEF
                <data android:mimeType="vnd.android.cursor.item/vnd
            </intent-filter>

            <intent-filter>
                <action android:name="android.intent.action.INSERT"
                <category android:name="android.intent.category.DEF
                <data android:mimeType="vnd.android.cursor.dir/vnd.
            </intent-filter>

        </activity>

        <activity class=".TitleEditor" android:label="@string/title
            android:theme="@android:style/Theme.Dialog">
            <intent-filter android:label="@string/resolve_title">
                <action android:name="com.android.notepad.action.ED
                <category android:name="android.intent.category.DEF
                <category android:name="android.intent.category.ALT
                <category android:name="android.intent.category.SEL
                <data android:mimeType="vnd.android.cursor.item/vnd
            </intent-filter>
        </activity>

    </application>
</manifest>
```

The first activity, `com.android.notepad.NotesList`, serves as our main entry into the app. It can do three things as described by its three intent templates:

1. 

```
<intent-filter>
  <action android:name="android.intent.action.MAIN" />
  <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

This provides a top-level entry into the NotePad application: the standard MAIN action is a main entry point (not requiring any other information in the Intent), and the LAUNCHER category says that this entry point should be listed in the application launcher.

2. 

```
<intent-filter>
  <action android:name="android.intent.action.VIEW" />
  <action android:name="android.intent.action.EDIT" />
  <action android:name="android.intent.action.PICK" />
  <category android:name="android.intent.category.DEFAULT" />
  <data mimeType="vnd.android.cursor.dir/vnd.google.note" />
</intent-filter>
```

This declares the things that the activity can do on a directory of notes. The type being supported is given with the `<type>` tag, where `vnd.android.cursor.dir/vnd.google.note` is a URI from which a Cursor of zero or more items (`vnd.android.cursor.dir`) can be retrieved which holds our note pad data (`vnd.google.note`). The activity allows the user to view or edit the directory of data (via the VIEW and EDIT actions), or to pick a particular note and return it to the caller (via the PICK action). Note also the DEFAULT category supplied here: this is required for the `Context.startActivity` ([/reference/android/content/Context.html#startActivity\(android.content.Intent\)](/reference/android/content/Context.html#startActivity(android.content.Intent))) method to resolve your activity when its component name is not explicitly specified.

3. 

```
<intent-filter>
  <action android:name="android.intent.action.GET_CONTENT" />
  <category android:name="android.intent.category.DEFAULT" />
  <data android:mimeType="vnd.android.cursor.item/vnd.google.note" />
</intent-filter>
```

This filter describes the ability return to the caller a note selected by the user without needing to know where it came from. The data type `vnd.android.cursor.item/vnd.google.note` is a URI from which a Cursor of exactly one (`vnd.android.cursor.item`) item can be retrieved which contains our note pad data (`vnd.google.note`). The GET\_CONTENT action is similar to the PICK action, where the activity will return to its caller a piece of data selected by the user. Here, however, the caller specifies the type of data they desire instead of the type of data the user will be picking from.

Given these capabilities, the following intents will resolve to the NotesList activity.

- **{ action=android.app.action.MAIN }** matches all of the activities that can be used as top-level entry points into an application.
- **{ action=android.app.action.MAIN, category=android.app.category.LAUNCHER }** is the actual intent used by the Launcher to populate its top-level list.
- **{ action=android.intent.action.VIEW data=content://com.google.provider.NotePad/notes }** displays a list of all the notes under "content://com.google.provider.NotePad/notes", which the user can browse through and see the details on.

- { **action=android.app.action.PICK** **data=content://com.google.provider.NotePad/notes** } provides a list of the notes under "content://com.google.provider.NotePad/notes", from which the user can pick a note whose data URL is returned back to the caller.
- { **action=android.app.action.GET\_CONTENT** **type=vnd.android.cursor.item/vnd.google.note** } is similar to the pick action, but allows the caller to specify the kind of data they want back so that the system can find the appropriate activity to pick something of that data type.

The second activity, `com.android.notepad.NoteEditor`, shows the user a single note entry and allows them to edit it. It can do two things as described by its two intent templates:

1. 

```
<intent-filter android:label="@string/resolve_edit">
  <action android:name="android.intent.action.VIEW" />
  <action android:name="android.intent.action.EDIT" />
  <category android:name="android.intent.category.DEFAULT" />
  <data android:mimeType="vnd.android.cursor.item/vnd.google.note" />
</intent-filter>
```

The first, primary, purpose of this activity is to let the user interact with a single note, as described by the MIME type `vnd.android.cursor.item/vnd.google.note`. The activity can either VIEW a note or allow the user to EDIT it. Again we support the DEFAULT category to allow the activity to be launched without explicitly specifying its component.

2. 

```
<intent-filter>
  <action android:name="android.intent.action.INSERT" />
  <category android:name="android.intent.category.DEFAULT" />
  <data android:mimeType="vnd.android.cursor.dir/vnd.google.note" />
</intent-filter>
```

The secondary use of this activity is to insert a new note entry into an existing directory of notes. This is used when the user creates a new note: the INSERT action is executed on the directory of notes, causing this activity to run and have the user create the new note data which it then adds to the content provider.

Given these capabilities, the following intents will resolve to the `NoteEditor` activity:

- { **action=android.intent.action.VIEW** **data=content://com.google.provider.NotePad/notes/{ID}** } shows the user the content of note {ID}.
- { **action=android.app.action.EDIT** **data=content://com.google.provider.NotePad/notes/{ID}** } allows the user to edit the content of note {ID}.
- { **action=android.app.action.INSERT** **data=content://com.google.provider.NotePad/notes** } creates a new, empty note in the notes list at "content://com.google.provider.NotePad/notes" and allows the user to edit it. If they keep their changes, the URI of the newly created note is returned to the caller.

The last activity, `com.android.notepad.TitleEditor`, allows the user to edit the title of a note. This could be implemented as a class that the application directly invokes (by explicitly setting its component in the Intent), but here we show a way you can publish alternative operations on existing data:

```
<intent-filter android:label="@string/resolve_title">
  <action android:name="com.android.notepad.action.EDIT_TITLE" />
  <category android:name="android.intent.category.DEFAULT" />
  <category android:name="android.intent.category.ALTERNATIVE" />
  <category android:name="android.intent.category.SELECTED_ALTERNATIVE" />
  <data android:mimeType="vnd.android.cursor.item/vnd.google.note" />
</intent-filter>
```

In the single intent template here, we have created our own private action called `com.android.notepad.action.EDIT_TITLE` which means to edit the title of a note. It must be invoked on a specific note (data type `vnd.android.cursor.item/vnd.google.note`) like the previous view and edit actions, but here displays and edits the title contained in the note data.

In addition to supporting the default category as usual, our title editor also supports two other standard categories: `ALTERNATIVE` and `SELECTED_ALTERNATIVE`. Implementing these categories allows others to find the special action it provides without directly knowing about it, through the `queryIntentActivityOptions(ComponentName, Intent[], Intent, int)` ([/reference/android/content/pm/PackageManager.html#queryIntentActivityOptions\(android.content.ComponentName, android.content.Intent\[\], android.content.Intent, int\)](#)) method, or more often to build dynamic menu items with `addIntentOptions(int, int, int, ComponentName, Intent[], Intent, int, MenuItem[])` ([/reference/android/view/Menu.html#addIntentOptions\(int, int, int, android.content.ComponentName, android.content.Intent\[\], android.content.Intent, int, android.view.MenuItem\[\]\)](#)). Note that in the intent template here was also supply an explicit name for the template (via `android:label="@string/resolve_title"`) to better control what the user sees when presented with this activity as an alternative action to the data they are viewing.

Given these capabilities, the following intent will resolve to the `TitleEditor` activity:

- **{ action=com.android.notepad.action.EDIT\_TITLE  
data=content://com.google.provider.NotePad/notes/{ID} }** displays and allows the user to edit the title associated with note `{ID}`.

### Standard Activity Actions

These are the current standard actions that Intent defines for launching activities (usually through `startActivity(Intent)` ([/reference/android/content/Context.html#startActivity\(android.content.Intent\)](#)). The most important, and by far most frequently used, are `ACTION_MAIN` ([/reference/android/content/Intent.html#ACTION\\_MAIN](#)) and `ACTION_EDIT` ([/reference/android/content/Intent.html#ACTION\\_EDIT](#)).

- [ACTION\\_MAIN](#)
- [ACTION\\_VIEW](#)
- [ACTION\\_ATTACH\\_DATA](#)
- [ACTION\\_EDIT](#)
- [ACTION\\_PICK](#)
- [ACTION\\_CHOOSER](#)
- [ACTION\\_GET\\_CONTENT](#)
- [ACTION\\_DIAL](#)
- [ACTION\\_CALL](#)
- [ACTION\\_SEND](#)
- [ACTION\\_SENDTO](#)
- [ACTION\\_ANSWER](#)
- [ACTION\\_INSERT](#)
- [ACTION\\_DELETE](#)
- [ACTION\\_RUN](#)
- [ACTION\\_SYNC](#)
- [ACTION\\_PICK\\_ACTIVITY](#)
- [ACTION\\_SEARCH](#)
- [ACTION\\_WEB\\_SEARCH](#)
- [ACTION\\_FACTORY\\_TEST](#)

### Standard Broadcast Actions

These are the current standard actions that Intent defines for receiving broadcasts (usually through `registerReceiver(BroadcastReceiver, IntentFilter)` ([/reference/android](#))).

[/content/Context.html#registerReceiver\(android.content.BroadcastReceiver, android.content.IntentFilter\)\)](#) or a <receiver> tag in a manifest).

- [ACTION\\_TIME\\_TICK](#)
- [ACTION\\_TIME\\_CHANGED](#)
- [ACTION\\_TIMEZONE\\_CHANGED](#)
- [ACTION\\_BOOT\\_COMPLETED](#)
- [ACTION\\_PACKAGE\\_ADDED](#)
- [ACTION\\_PACKAGE\\_CHANGED](#)
- [ACTION\\_PACKAGE\\_REMOVED](#)
- [ACTION\\_PACKAGE\\_RESTARTED](#)
- [ACTION\\_PACKAGE\\_DATA\\_CLEARED](#)
- [ACTION\\_UID\\_REMOVED](#)
- [ACTION\\_BATTERY\\_CHANGED](#)
- [ACTION\\_POWER\\_CONNECTED](#)
- [ACTION\\_POWER\\_DISCONNECTED](#)
- [ACTION\\_SHUTDOWN](#)

### Standard Categories

These are the current standard categories that can be used to further clarify an Intent via [addCategory\(String\)](#) ([/reference/android/content/Intent.html#addCategory\(java.lang.String\)](#)).

- [CATEGORY\\_DEFAULT](#)
- [CATEGORY\\_BROWSABLE](#)
- [CATEGORY\\_TAB](#)
- [CATEGORY\\_ALTERNATIVE](#)
- [CATEGORY\\_SELECTED\\_ALTERNATIVE](#)
- [CATEGORY\\_LAUNCHER](#)
- [CATEGORY\\_INFO](#)
- [CATEGORY\\_HOME](#)
- [CATEGORY\\_PREFERENCE](#)
- [CATEGORY\\_TEST](#)
- [CATEGORY\\_CAR\\_DOCK](#)
- [CATEGORY\\_DESK\\_DOCK](#)
- [CATEGORY\\_LE\\_DESK\\_DOCK](#)
- [CATEGORY\\_HE\\_DESK\\_DOCK](#)
- [CATEGORY\\_CAR\\_MODE](#)
- [CATEGORY\\_APP\\_MARKET](#)

### Standard Extra Data

These are the current standard fields that can be used as extra data via [putExtra\(String, Bundle\)](#) ([/reference/android/content/Intent.html#putExtra\(java.lang.String, android.os.Bundle\)](#)).

- [EXTRA\\_ALARM\\_COUNT](#)
- [EXTRA\\_BCC](#)
- [EXTRA\\_CC](#)
- [EXTRA\\_CHANGED\\_COMPONENT\\_NAME](#)
- [EXTRA\\_DATA\\_REMOVED](#)
- [EXTRA\\_DOCK\\_STATE](#)
- [EXTRA\\_DOCK\\_STATE\\_HE\\_DESK](#)
- [EXTRA\\_DOCK\\_STATE\\_LE\\_DESK](#)
- [EXTRA\\_DOCK\\_STATE\\_CAR](#)
- [EXTRA\\_DOCK\\_STATE\\_DESK](#)
- [EXTRA\\_DOCK\\_STATE\\_UNDOCKED](#)
- [EXTRA\\_DONT\\_KILL\\_APP](#)



- [EXTRA\\_EMAIL](#)
- [EXTRA\\_INITIAL\\_INTENTS](#)
- [EXTRA\\_INTENT](#)
- [EXTRA\\_KEY\\_EVENT](#)
- [EXTRA\\_ORIGINATING\\_URI](#)
- [EXTRA\\_PHONE\\_NUMBER](#)
- [EXTRA\\_REFERRER](#)
- [EXTRA\\_REMOTE\\_INTENT\\_TOKEN](#)
- [EXTRA\\_REPLACING](#)
- [EXTRA\\_SHORTCUT\\_ICON](#)
- [EXTRA\\_SHORTCUT\\_ICON\\_RESOURCE](#)
- [EXTRA\\_SHORTCUT\\_INTENT](#)
- [EXTRA\\_STREAM](#)
- [EXTRA\\_SHORTCUT\\_NAME](#)
- [EXTRA\\_SUBJECT](#)
- [EXTRA\\_TEMPLATE](#)
- [EXTRA\\_TEXT](#)
- [EXTRA\\_TITLE](#)
- [EXTRA\\_UID](#)

## Flags

These are the possible flags that can be used in the Intent via [setFlags\(int\)](#) ([//reference/android/content/Intent.html#setFlags\(int\)](#)) and [addFlags\(int\)](#) ([//reference/android/content/Intent.html#addFlags\(int\)](#)). See [setFlags\(int\)](#) ([//reference/android/content/Intent.html#setFlags\(int\)](#)) for a list of all possible flags.

## Summary

---

### Nested Classes

- class Intent.FilterComparison      Wrapper class holding an Intent and implementing comparisons on it for the purpose of filtering.
- class Intent.ShortcutIconResource      Represents a shortcut/live folder icon resource.

### Constants

String ACTION_AIRPLANE_MODE_CHANGED	Broadcast Action: The user has switched the phone into on or out of Airplane Mode.
String ACTION_ALL_APPS	Activity Action: List all available applications Input: Nothing.
String ACTION_ANSWER	Activity Action: Handle an incoming phone call.
String ACTION_APP_ERROR	Activity Action: The user pressed the "Report" button in the crash/ANR dialog.
String ACTION_ASSIST	Activity Action: Perform assist action.
String ACTION_ATTACH_DATA	Used to indicate that some piece of data should be attached to some other place.
String ACTION_BATTERY_CHANGED	Broadcast Action: This is a <i>sticky broadcast</i> containing the charging state, level, and other information about the battery.
String ACTION_BATTERY_LOW	Broadcast Action: Indicates low battery condition on the device.
String ACTION_BATTERY_OKAY	Broadcast Action: Indicates the battery is now okay after being low.
String ACTION_BOOT_COMPLETED	Broadcast Action: This is broadcast once, after the system has finished booting.
String ACTION_BUG_REPORT	Activity Action: Show activity for reporting a bug.

String ACTION_CALL	Activity Action: Perform a call to someone specified by the data.
String ACTION_CALL_BUTTON	Activity Action: The user pressed the "call" button to go to dialer or other appropriate UI for placing a call.
String ACTION_CAMERA_BUTTON	Broadcast Action: The "Camera Button" was pressed.
String ACTION_CHOOSER	Activity Action: Display an activity chooser, allowing the user to pick what they want to before proceeding.
String ACTION_CLOSE_SYSTEM_DIALOGS	Broadcast Action: This is broadcast when a user action should request a temporary system dialog to dismiss.
String ACTION_CONFIGURATION_CHANGED	Broadcast Action: The current device Configuration (orientation, locale, etc) has changed.
String ACTION_CREATE_DOCUMENT	Activity Action: Allow the user to create a new document.
String ACTION_CREATE_SHORTCUT	Activity Action: Creates a shortcut.
String ACTION_DATE_CHANGED	Broadcast Action: The date has changed.
String ACTION_DEFAULT	A synonym for ACTION_VIEW, the "standard" action that is performed on a piece of data.
String ACTION_DELETE	Activity Action: Delete the given data from its container. Broadcast Action: A sticky broadcast that indicates low memory condition on the device
String ACTION_DEVICE_STORAGE_LOW	<div>This is a protected intent that can only be sent by the system.</div> Broadcast Action: Indicates low memory condition on the device no longer exists
String ACTION_DEVICE_STORAGE_OK	<div>This is a protected intent that can only be sent by the system.</div>
String ACTION_DIAL	Activity Action: Dial a number as specified by the data.
String ACTION_DOCK_EVENT	Broadcast Action: A sticky broadcast for changes in the physical docking state of the device.
String ACTION_DREAMING_STARTED	Broadcast Action: Sent after the system starts dreaming.
String ACTION_DREAMING_STOPPED	Broadcast Action: Sent after the system stops dreaming.
String ACTION_EDIT	Activity Action: Provide explicit editable access to the given data.
String ACTION_EXTERNAL_APPLICATIONS_AVAILABLE	Broadcast Action: Resources for a set of packages (which were previously unavailable) are currently available since the media on which they exist is available.
String ACTION_EXTERNAL_APPLICATIONS_UNAVAILABLE	Broadcast Action: Resources for a set of packages are currently unavailable since the media on which they exist is unavailable.
String ACTION_FACTORY_TEST	Activity Action: Main entry point for factory tests.
String ACTION_GET_CONTENT	Activity Action: Allow the user to select a particular kind of data and return it.
String ACTION_GET_RESTRICTION_ENTRIES	Broadcast to a specific application to query any supported restrictions to impose on restricted users.
String ACTION_GTALK_SERVICE_CONNECTED	Broadcast Action: A GTalk connection has been established.
String ACTION_GTALK_SERVICE_DISCONNECTED	Broadcast Action: A GTalk connection has been disconnected.
String ACTION_HEADSET_PLUG	Broadcast Action: Wired Headset plugged in or unplugged
String ACTION_INPUT_METHOD_CHANGED	Broadcast Action: An input method has been changed.
String ACTION_INSERT	Activity Action: Insert an empty item into the given container.
String ACTION_INSERT_OR_EDIT	Activity Action: Pick an existing item, or insert a new item, and then edit it.
String ACTION_INSTALL_PACKAGE	Activity Action: Launch application installer.

String ACTION_LOCALE_CHANGED	Broadcast Action: The current device's locale has changed.
String ACTION_MAIN	Activity Action: Start as a main entry point, does not expect to receive data.
String ACTION_MANAGE_NETWORK_USAGE	Activity Action: Show settings for managing network data usage of a specific application.
String ACTION_MANAGE_PACKAGE_STORAGE	Broadcast Action: Indicates low memory condition notification acknowledged by user and package manager should be started.
String ACTION_MEDIA_BAD_REMOVAL	Broadcast Action: External media was removed from SD card slot, but mount point was not unmounted.
String ACTION_MEDIA_BUTTON	Broadcast Action: The "Media Button" was pressed.
String ACTION_MEDIA_CHECKING	Broadcast Action: External media is present, and being disk-checked. The path to the mount point for the checking media is contained in the Intent.mData field.
String ACTION_MEDIA_EJECT	Broadcast Action: User has expressed the desire to remove the external storage media.
String ACTION_MEDIA_MOUNTED	Broadcast Action: External media is present and mounted its mount point.
String ACTION_MEDIA_NOFS	Broadcast Action: External media is present, but is using an incompatible fs (or is blank). The path to the mount point for the checking media is contained in the Intent.mData field.
String ACTION_MEDIA_REMOVED	Broadcast Action: External media has been removed.
String ACTION_MEDIA_SCANNER_FINISHED	Broadcast Action: The media scanner has finished scanning a directory.
String ACTION_MEDIA_SCANNER_SCAN_FILE	Broadcast Action: Request the media scanner to scan a file and add it to the media database.
String ACTION_MEDIA_SCANNER_STARTED	Broadcast Action: The media scanner has started scanning directory.
String ACTION_MEDIA_SHARED	Broadcast Action: External media is unmounted because it is being shared via USB mass storage.
String ACTION_MEDIA_UNMOUNTABLE	Broadcast Action: External media is present but cannot be mounted.
String ACTION_MEDIA_UNMOUNTED	Broadcast Action: External media is present, but not mounted at its mount point.
String ACTION_MY_PACKAGE_REPLACED	Broadcast Action: A new version of your application has been installed over an existing one.
String ACTION_NEW_OUTGOING_CALL	Broadcast Action: An outgoing call is about to be placed.
String ACTION_OPEN_DOCUMENT	Activity Action: Allow the user to select and return one or more existing documents.
String ACTION_PACKAGE_ADDED	Broadcast Action: A new application package has been installed on the device.
String ACTION_PACKAGE_CHANGED	Broadcast Action: An existing application package has been changed (e.g.
String ACTION_PACKAGE_DATA_CLEARED	Broadcast Action: The user has cleared the data of a package.
String ACTION_PACKAGE_FIRST_LAUNCH	Broadcast Action: Sent to the installer package of an application when that application is first launched (that is first time it is moved out of the stopped state).
String ACTION_PACKAGE_FULLY_REMOVED	Broadcast Action: An existing application package has been completely removed from the device.
String ACTION_PACKAGE_INSTALL	<i>This constant was deprecated in API level 14. This constant has never been used.</i>
String ACTION_PACKAGE_NEEDS_VERIFICATION	Broadcast Action: Sent to the system package verifier when a package needs to be verified.
String ACTION_PACKAGE_REMOVED	Broadcast Action: An existing application package has been removed from the device.

String ACTION_PACKAGE_REPLACED	Broadcast Action: A new version of an application package has been installed, replacing an existing version that was previously installed.
String ACTION_PACKAGE_RESTARTED	Broadcast Action: The user has restarted a package, and one or more of its processes have been killed.
String ACTION_PACKAGE_VERIFIED	Broadcast Action: Sent to the system package verifier when a package is verified.
String ACTION_PASTE	Activity Action: Create a new item in the given container, initializing it from the current contents of the clipboard.
String ACTION_PICK	Activity Action: Pick an item from the data, returning what was selected.
String ACTION_PICK_ACTIVITY	Activity Action: Pick an activity given an intent, returning the class selected.
String ACTION_POWER_CONNECTED	Broadcast Action: External power has been connected to the device.
String ACTION_POWER_DISCONNECTED	Broadcast Action: External power has been removed from the device.
String ACTION_POWER_USAGE_SUMMARY	Activity Action: Show power usage information to the user.
String ACTION_PROVIDER_CHANGED	Broadcast Action: Some content providers have parts of the namespace where they publish new events or items that the user may be especially interested in.
String ACTION_QUICK_CLOCK	Sent when the user taps on the clock widget in the system "quick settings" area.
String ACTION_REBOOT	Broadcast Action: Have the device reboot.
String ACTION_RUN	Activity Action: Run the data, whatever that means.
String ACTION_SCREEN_OFF	Broadcast Action: Sent after the screen turns off.
String ACTION_SCREEN_ON	Broadcast Action: Sent after the screen turns on.
String ACTION_SEARCH	Activity Action: Perform a search.
String ACTION_SEARCH_LONG_PRESS	Activity Action: Start action associated with long pressing the search key.
String ACTION_SEND	Activity Action: Deliver some data to someone else.
String ACTION_SENDTO	Activity Action: Send a message to someone specified by data.
String ACTION_SEND_MULTIPLE	Activity Action: Deliver multiple data to someone else.
String ACTION_SET_WALLPAPER	Activity Action: Show settings for choosing wallpaper.
String ACTION_SHUTDOWN	Input: Nothing.
String ACTION_SYNC	Broadcast Action: Device is shutting down.
String ACTION_SYSTEM_TUTORIAL	Activity Action: Perform a data synchronization.
String ACTION_TIMEZONE_CHANGED	Activity Action: Start the platform-defined tutorial.
String ACTION_TIME_CHANGED	Input: <code>getStringExtra(SearchManager.QUERY)</code> ( <a href="/reference/android/app/SearchManager.html#QUERY">/reference/android/app/SearchManager.html#QUERY</a> ) is the text search for.
String ACTION_TIME_TICK	Broadcast Action: The timezone has changed.
String ACTION_UID_REMOVED	Broadcast Action: The time was set.
String ACTION_UMS_CONNECTED	Broadcast Action: The current time has changed.
String ACTION_UMS_DISCONNECTED	Broadcast Action: A user ID has been removed from the system.
String ACTION_UNINSTALL_PACKAGE	<i>This constant was deprecated in API level 14. replaced by <code>android.os.storage.StorageEventListener</code></i>
	<i>This constant was deprecated in API level 14. replaced by <code>android.os.storage.StorageEventListener</code></i>
	Activity Action: Launch application uninstaller.

String ACTION_USER_BACKGROUND	Sent when a user switch is happening, causing the process user to be sent to the background.
String ACTION_USER_FOREGROUND	Sent when a user switch is happening, causing the process user to be brought to the foreground.
String ACTION_USER_INITIALIZE	Sent the first time a user is starting, to allow system apps perform one time initialization.
String ACTION_USER_PRESENT	Broadcast Action: Sent when the user is present after device wakes up (e.g when the keyguard is gone).
String ACTION_VIEW	Activity Action: Display the data to the user.
String ACTION_VOICE_COMMAND	Activity Action: Start Voice Command.
	<i>This constant was deprecated in API level 16. Modern applications should use <code>WindowManager.LayoutParams.FLAG_SHOW_WALLPAPER</code> to have the wallpaper shown behind their UI, rather than watching for this broadcast and rendering the wallpaper on their own.</i>
String ACTION_WALLPAPER_CHANGED	
String ACTION_WEB_SEARCH	Activity Action: Perform a web search.
String CATEGORY_ALTERNATIVE	Set if the activity should be considered as an alternative action to the data the user is currently viewing.
String CATEGORY_APP_BROWSER	Used with ACTION_MAIN to launch the browser application.
String CATEGORY_APP_CALCULATOR	Used with ACTION_MAIN to launch the calculator application.
String CATEGORY_APP_CALENDAR	Used with ACTION_MAIN to launch the calendar application.
String CATEGORY_APP_CONTACTS	Used with ACTION_MAIN to launch the contacts application.
String CATEGORY_APP_EMAIL	Used with ACTION_MAIN to launch the email application.
String CATEGORY_APP_GALLERY	Used with ACTION_MAIN to launch the gallery application.
String CATEGORY_APP_MAPS	Used with ACTION_MAIN to launch the maps application.
String CATEGORY_APP_MARKET	This activity allows the user to browse and download new applications.
String CATEGORY_APP_MESSAGING	Used with ACTION_MAIN to launch the messaging application.
String CATEGORY_APP_MUSIC	Used with ACTION_MAIN to launch the music application.
String CATEGORY_BROWSABLE	Activities that can be safely invoked from a browser must support this category.
String CATEGORY_CAR_DOCK	An activity to run when device is inserted into a car dock.
String CATEGORY_CAR_MODE	Used to indicate that the activity can be used in a car environment.
String CATEGORY_DEFAULT	Set if the activity should be an option for the default action (center press) to perform on a piece of data.
String CATEGORY_DESK_DOCK	An activity to run when device is inserted into a car dock.
String CATEGORY_DEVELOPMENT_PREFERENCE	This activity is a development preference panel.
String CATEGORY_EMBED	Capable of running inside a parent activity container.
String CATEGORY_FRAMEWORK_INSTRUMENTATION_TEST	To be used as code under test for framework instrumentation tests.
String CATEGORY_HE_DESK_DOCK	An activity to run when device is inserted into a digital (high end) dock.
String CATEGORY_HOME	This is the home activity, that is the first activity that is displayed when the device boots.
String CATEGORY_INFO	Provides information about the package it is in; typically used if a package does not contain a CATEGORY_LAUNCHER to provide a front-door to the user without having to be shown the all apps list.
String CATEGORY_LAUNCHER	Should be displayed in the top-level launcher.

String CATEGORY_LE_DESK DOCK	An activity to run when device is inserted into a analog (lo end) dock.
String CATEGORY_MONKEY	This activity may be exercised by the monkey or other automated test tools.
String CATEGORY_OPENABLE	Used to indicate that an intent only wants URIs that can b opened with <code>openFileDescriptor(Uri, String)</code> .
String CATEGORY_PREFERENCE	This activity is a preference panel.
String CATEGORY_SAMPLE_CODE	To be used as a sample code example (not part of the nori user experience).
String CATEGORY_SELECTED_ALTERNATIVE	Set if the activity should be considered as an alternative selection action to the data the user has currently selecte
String CATEGORY_TAB	Intended to be used as a tab inside of a containing TabActivity.
String CATEGORY_TEST	To be used as a test (not part of the normal user experienc
String CATEGORY_UNIT_TEST	To be used as a unit test (run through the Test Harness).
String EXTRA_ALARM_COUNT	Used as an int extra field in <code>AlarmManager</code> intents to tell application being invoked how many pending alarms are being delievered with the intent.
String EXTRA_ALLOW_MULTIPLE	Extra used to indicate that an intent can allow the user to select and return multiple items. <i>This constant was deprecated in API level 16. As of JELLY_BEAN, Android will no longer show an interstitial message about updating existing applications so this is no longer needed.</i>
String EXTRA_ALLOW_REPLACE	
String EXTRA_ASSIST_CONTEXT	An optional field on <code>ACTION_ASSIST</code> and containing additional contextual information supplied by the current foreground app at the time of the assist request.
String EXTRA_ASSIST_PACKAGE	An optional field on <code>ACTION_ASSIST</code> containing the name the current foreground application package at the time the assist was invoked.
String EXTRA_BCC	A <code>String[]</code> holding e-mail addresses that should be blind carbon copied.
String EXTRA_BUG_REPORT	Used as a parcelable extra field in <code>ACTION_APP_ERROR</code> , containing the bug report.
String EXTRA_CC	A <code>String[]</code> holding e-mail addresses that should be carbon copied.
String EXTRA_CHANGED_COMPONENT_NAME	<i>This constant was deprecated in API level 7. See EXTRA_CHANGED_COMPONENT_NAME_LIST; this field will contain only the first name in the list.</i>
String EXTRA_CHANGED_COMPONENT_NAME_LIST	This field is part of <code>ACTION_PACKAGE_CHANGED</code> , and contains a string array of all of the components that have changed. This field is part of
String EXTRA_CHANGED_PACKAGE_LIST	<code>ACTION_EXTERNAL_APPLICATIONS_AVAILABLE</code> , <code>ACTION_EXTERNAL_APPLICATIONS_UNAVAILABLE</code> and contains a string array of all of the components that have changed. This field is part of
String EXTRA_CHANGED_UID_LIST	<code>ACTION_EXTERNAL_APPLICATIONS_AVAILABLE</code> , <code>ACTION_EXTERNAL_APPLICATIONS_UNAVAILABLE</code> and contains an integer array of uids of all of the components that have changed.
String EXTRA_DATA_REMOVED	Used as a boolean extra field in <code>ACTION_PACKAGE_REMO</code> intents to indicate whether this represents a full uninstall (removing both the code and its data) or a partial uninstall (leaving its data, implying that this is an update).

String EXTRA_DOCK_STATE	Used as an int extra field in ACTION_DOCK_EVENT intents request the dock state.
int EXTRA_DOCK_STATE_CAR	Used as an int value for EXTRA_DOCK_STATE to represent that the phone is in a car dock.
int EXTRA_DOCK_STATE_DESK	Used as an int value for EXTRA_DOCK_STATE to represent that the phone is in a desk dock.
int EXTRA_DOCK_STATE_HE_DESK	Used as an int value for EXTRA_DOCK_STATE to represent that the phone is in a digital (high end) dock.
int EXTRA_DOCK_STATE_LE_DESK	Used as an int value for EXTRA_DOCK_STATE to represent that the phone is in a analog (low end) dock.
int EXTRA_DOCK_STATE_UNDOCKED	Used as an int value for EXTRA_DOCK_STATE to represent that the phone is not in any dock.
String EXTRA_DONT_KILL_APP	Used as a boolean extra field in ACTION_PACKAGE_REMOVE or ACTION_PACKAGE_CHANGED intents to override the default action of restarting the application.
String EXTRA_EMAIL	A String[] holding e-mail addresses that should be delivered to.
String EXTRA_HTML_TEXT	A constant String that is associated with the Intent, used with ACTION_SEND to supply an alternative to EXTRA_TEXT as HTML formatted text.
String EXTRA_INITIAL_INTENTS	A Parcelable[] of Intent or LabeledIntent objects as supplied with putExtra(String, Parcelable[]) of additional activities to place at the front of the list of choices, when shown to the user with a ACTION_CHOOSER.
String EXTRA_INSTALLER_PACKAGE_NAME	Used as a string extra field with ACTION_INSTALL_PACKAGE to install a package.
String EXTRA_INTENT	An Intent describing the choices you would like shown with ACTION_PICK_ACTIVITY.
String EXTRA_KEY_EVENT	A KeyEvent object containing the event that triggered the creation of the Intent it is in.
String EXTRA_LOCAL_ONLY	Extra used to indicate that an intent should only return data that is on the local device.
String EXTRA_MIME_TYPES	Extra used to communicate a set of acceptable MIME types.
String EXTRA_NOT_UNKNOWN_SOURCE	Used as a boolean extra field with ACTION_INSTALL_PACKAGE to install a package.
String EXTRA_ORIGINATING_URI	Used as a URI extra field with ACTION_INSTALL_PACKAGE and ACTION_VIEW to indicate the URI from which the local APK in the Intent data field originated from.
String EXTRA_PHONE_NUMBER	A String holding the phone number originally entered in ACTION_NEW_OUTGOING_CALL, or the actual number to call in a ACTION_CALL.
String EXTRA_REFERRER	Used as a URI extra field with ACTION_INSTALL_PACKAGE and ACTION_VIEW to indicate the HTTP referrer URI associated with the Intent data field or EXTRA_ORIGINATING_URI.
String EXTRA_REMOTE_INTENT_TOKEN	Used in the extra field in the remote intent.
String EXTRA_REPLACING	Used as a boolean extra field in ACTION_PACKAGE_REMOVE or ACTION_PACKAGE_CHANGED intents to indicate that this is a replacement of the package so this broadcast will immediately be followed by an add broadcast for a different version of the same package.
String EXTRA_RESTRICTIONS_BUNDLE	Extra sent in the intent to the BroadcastReceiver that handles ACTION_GET_RESTRICTION_ENTRIES.
String EXTRA_RESTRICTIONS_INTENT	Extra used in the response from a BroadcastReceiver that handles ACTION_GET_RESTRICTION_ENTRIES.

String EXTRA_RESTRICTIONS_LIST	Extra used in the response from a BroadcastReceiver that handles ACTION_GET_RESTRICTION_ENTRIES.
String EXTRA_RETURN_RESULT	Used as a boolean extra field with ACTION_INSTALL_PACKAGE or ACTION_UNINSTALL_PACKAGE.
String EXTRA_SHORTCUT_ICON	The name of the extra used to define the icon, as a Bitmap a shortcut.
String EXTRA_SHORTCUT_ICON_RESOURCE	The name of the extra used to define the icon, as a ShortcutIconResource, of a shortcut.
String EXTRA_SHORTCUT_INTENT	The name of the extra used to define the Intent of a shortcut.
String EXTRA_SHORTCUT_NAME	The name of the extra used to define the name of a shortcut.
String EXTRA_SHUTDOWN_USERSPACE_ONLY	Optional extra for ACTION_SHUTDOWN that allows the sender to qualify that this shutdown is only for the user space of the system, not a complete shutdown.
String EXTRA_STREAM	A content: URI holding a stream of data associated with the Intent, used with ACTION_SEND to supply the data being sent.
String EXTRA_SUBJECT	A constant string holding the desired subject line of a message.
String EXTRA_TEMPLATE	The initial data to place in a newly created record.
String EXTRA_TEXT	A constant CharSequence that is associated with the Intent used with ACTION_SEND to supply the literal data to be sent.
String EXTRA_TITLE	A CharSequence dialog title to provide to the user when used with a ACTION_CHOOSER.
String EXTRA_UID	Used as an int extra field in ACTION_UID_REMOVED intent to supply the uid the package had been assigned.
int FILL_IN_ACTION	Use with fillIn(Intent, int) to allow the current action value to be overwritten, even if it is already set.
int FILL_IN_CATEGORIES	Use with fillIn(Intent, int) to allow the current categories to be overwritten, even if they are already set.
int FILL_IN_CLIP_DATA	Use with fillIn(Intent, int) to allow the current ClipData to be overwritten, even if it is already set.
int FILL_IN_COMPONENT	Use with fillIn(Intent, int) to allow the current component value to be overwritten, even if it is already set.
int FILL_IN_DATA	Use with fillIn(Intent, int) to allow the current data or type value overwritten, even if it is already set.
int FILL_IN_PACKAGE	Use with fillIn(Intent, int) to allow the current package value to be overwritten, even if it is already set.
int FILL_IN_SELECTOR	Use with fillIn(Intent, int) to allow the current selector to be overwritten, even if it is already set.
int FILL_IN_SOURCE_BOUNDS	Use with fillIn(Intent, int) to allow the current bounds rectangle to be overwritten, even if it is already set.
int FLAG_ACTIVITY_BROUGHT_TO_FRONT	This flag is not normally set by application code, but set for you by the system as described in the launchMode documentation for the singleTask mode.
int FLAG_ACTIVITY_CLEAR_TASK	If set in an Intent passed to Context.startActivity(), this flag will cause any existing task that would be associated with the activity to be cleared before the activity is started.
int FLAG_ACTIVITY_CLEAR_TOP	If set, and the activity being launched is already running in the current task, then instead of launching a new instance of that activity, all of the other activities on top of it will be closed and this Intent will be delivered to the (now on top) activity as a new Intent.



<code>int FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET</code>	If set, this marks a point in the task's activity stack that should be cleared when the task is reset.
<code>int FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS</code>	If set, the new activity is not kept in the list of recently launched activities.
<code>int FLAG_ACTIVITY_FORWARD_RESULT</code>	If set and this intent is being used to launch a new activity from an existing one, then the reply target of the existing activity will be transferred to the new activity.
<code>int FLAG_ACTIVITY_LAUNCHED_FROM_HISTORY</code>	This flag is not normally set by application code, but set for you by the system if this activity is being launched from history (longpress home key).
<code>int FLAG_ACTIVITY_MULTIPLE_TASK</code>	<b>Do not use this flag unless you are implementing your own top-level application launcher.</b>
<code>int FLAG_ACTIVITY_NEW_TASK</code>	If set, this activity will become the start of a new task on the history stack.
<code>int FLAG_ACTIVITY_NO_ANIMATION</code>	If set in an Intent passed to <code>Context.startActivity()</code> , this flag will prevent the system from applying an activity transition animation to go to the next activity state.
<code>int FLAG_ACTIVITY_NO_HISTORY</code>	If set, the new activity is not kept in the history stack.
<code>int FLAG_ACTIVITY_NO_USER_ACTION</code>	If set, this flag will prevent the normal <code>onUserLeaveHint</code> callback from occurring on the current frontmost activity before it is paused as the newly-started activity is brought to the front.
<code>int FLAG_ACTIVITY_PREVIOUS_IS_TOP</code>	If set and this intent is being used to launch a new activity from an existing one, 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 one.
<code>int FLAG_ACTIVITY_REORDER_TO_FRONT</code>	If set in an Intent passed to <code>Context.startActivity()</code> , this flag will cause the launched activity to be brought to the front of its task's history stack if it is already running.
<code>int FLAG_ACTIVITY_RESET_TASK_IF_NEEDED</code>	If set, and this activity is either being started in a new task bringing to the top an existing task, then it will be launched as the front door of the task.
<code>int FLAG_ACTIVITY_SINGLE_TOP</code>	If set, the activity will not be launched if it is already running at the top of the history stack.
<code>int FLAG_ACTIVITY_TASK_ON_HOME</code>	If set in an Intent passed to <code>Context.startActivity()</code> , this flag will cause a newly launching task to be placed on top of the current home activity task (if there is one).
<code>int FLAG_DEBUG_LOG_RESOLUTION</code>	A flag you can enable for debugging: when set, log messages will be printed during the resolution of this intent to show what has been found to create the final resolved list.
<code>int FLAG_EXCLUDE_STOPPED_PACKAGES</code>	If set, this intent will not match any components in packages that are currently stopped.
<code>int FLAG_FROM_BACKGROUND</code>	Can be set by the caller to indicate that this Intent is coming from a background operation, not from direct user interaction.
<code>int FLAG_GRANT_PERSISTABLE_URI_PERMISSION</code>	When combined with <code>FLAG_GRANT_READ_URI_PERMISSION</code> and/or <code>FLAG_GRANT_WRITE_URI_PERMISSION</code> , the URI permission grant can be persisted across device reboots until explicitly revoked with <code>revokeUriPermission(Uri, int)</code> .
<code>int FLAG_GRANT_READ_URI_PERMISSION</code>	If set, the recipient of this Intent will be granted permission to perform read operations on the URI in the Intent's data and any URIs specified in its <code>ClipData</code> .
<code>int FLAG_GRANT_WRITE_URI_PERMISSION</code>	If set, the recipient of this Intent will be granted permission to perform write operations on the URI in the Intent's data and any URIs specified in its <code>ClipData</code> .

int FLAG_INCLUDE_STOPPED_PACKAGES	If set, this intent will always match any components in packages that are currently stopped.
int FLAG_RECEIVER_FOREGROUND	If set, when sending a broadcast the recipient is allowed to run at foreground priority, with a shorter timeout interval.
int FLAG_RECEIVER_NO_ABORT	If this is an ordered broadcast, don't allow receivers to abort the broadcast.
int FLAG_RECEIVER_REGISTERED_ONLY	If set, when sending a broadcast only registered receivers be called – no BroadcastReceiver components will be launched.
int FLAG_RECEIVER_REPLACE_PENDING	If set, when sending a broadcast the new broadcast will replace any existing pending broadcast that matches it.
String METADATA_DOCK_HOME	Boolean that can be supplied as meta-data with a dock activity, to indicate that the dock should take over the home key when it is active.
int URI_INTENT_SCHEME	Flag for use with <code>toUri(int)</code> and <code>parseUri(String, int)</code> : the URI string always has the "intent:" scheme.

**Inherited Constants** [Expand]► From interface `android.os.Parcelable`**Fields**`public static final Creator<Intent> CREATOR`**Public Constructors**

`Intent()`  
Create an empty intent.

`Intent(Intent o)`  
Copy constructor.

`Intent(String action)`  
Create an intent with a given action.

`Intent(String action, Uri uri)`  
Create an intent with a given action and for a given data url.

`Intent(Context packageContext, Class<?> cls)`  
Create an intent for a specific component.

`Intent(String action, Uri uri, Context packageContext, Class<?> cls)`  
Create an intent for a specific component with a specified action and data.

**Public Methods**

`Intent addCategory(String category)`  
Add a new category to the intent.

`Intent addFlags(int flags)`  
Add additional flags to the intent (or with existing flags value).

`Object clone()`  
Creates and returns a copy of this `Object`.

`cloneFilter()`  
Make a clone of only the parts of the `Intent` that are relevant for filter matching: the action, data, type, component, and categories.

`static Intent createChooser(Intent target, CharSequence title)`  
Convenience function for creating a `ACTION_CHOOSER` `Intent`.

`describeContents()`

`int` `describeContents()`  
Describe the kinds of special objects contained in this `Parcelable`'s marshalled representation.

`fillIn(Intent other, int flags)`  
Copy the contents of *other* in to this object, but only where fields are not defined by this object.

`filterEquals(Intent other)`

`boolean` `filterEquals(Intent other)`  
Determine if two intents are the same for the purposes of intent resolution

```

        (filtering).
    int filterHashCode()
        Generate hash code that matches semantics of filterEquals().
    String getAction()
        Retrieve the general action to be performed, such as ACTION_VIEW.
    boolean[] getBooleanArrayExtra (String name)
        Retrieve extended data from the intent.
    boolean getBooleanExtra (String name, boolean defaultValue)
        Retrieve extended data from the intent.
    Bundle getBundleExtra (String name)
        Retrieve extended data from the intent.
    byte[] getByteArrayExtra (String name)
        Retrieve extended data from the intent.
    byte getByteExtra (String name, byte defaultValue)
        Retrieve extended data from the intent.
    Set<String> getCategories ()
        Return the set of all categories in the intent.
    char[] getCharArrayExtra (String name)
        Retrieve extended data from the intent.
    char getCharExtra (String name, char defaultValue)
        Retrieve extended data from the intent.
    CharSequence[] getCharSequenceArrayExtra (String name)
        Retrieve extended data from the intent.
    ArrayList<CharSequence> getCharSequenceArrayListExtra (String name)
        Retrieve extended data from the intent.
    CharSequence getCharSequenceExtra (String name)
        Retrieve extended data from the intent.
    ClipData getClipData ()
        Return the ClipData associated with this Intent.
    ComponentName getComponent ()
        Retrieve the concrete component associated with the intent.
    Uri getData ()
        Retrieve data this intent is operating on.
    String getDataString ()
        The same as getData ( ), but returns the URI as an encoded String.
    double[] getDoubleArrayExtra (String name)
        Retrieve extended data from the intent.
    double getDoubleExtra (String name, double defaultValue)
        Retrieve extended data from the intent.
    Bundle getExtras ()
        Retrieves a map of extended data from the intent.
    int getFlags ()
        Retrieve any special flags associated with this intent.
    float[] getFloatArrayExtra (String name)
        Retrieve extended data from the intent.
    float getFloatExtra (String name, float defaultValue)
        Retrieve extended data from the intent.
    int[] getIntArrayExtra (String name)
        Retrieve extended data from the intent.
    int getIntExtra (String name, int defaultValue)
        Retrieve extended data from the intent.

```

```

    ArrayList<Integer> getIntegerArrayListExtra (String name)
        Retrieve extended data from the intent.
    getIntent (String uri)
    static Intent This method was deprecated in API level 4. Use parseUri (String, int) instead.
    static Intent getIntentOld (String uri)
    long[] getLongArrayExtra (String name)
        Retrieve extended data from the intent.
    long getLongExtra (String name, long defaultValue)
        Retrieve extended data from the intent.
    String getPackage ()
        Retrieve the application package name this Intent is limited to.
    Parcelable[] getParcelableArrayExtra (String name)
        Retrieve extended data from the intent.
    <T extends Parcelable> ArrayList<T> getParcelableArrayListExtra (String name)
        Retrieve extended data from the intent.
    <T extends Parcelable> T getParcelableExtra (String name)
        Retrieve extended data from the intent.
    String getScheme ()
        Return the scheme portion of the intent's data.
    Intent getSelector ()
        Return the specific selector associated with this Intent.
    Serializable getSerializableExtra (String name)
        Retrieve extended data from the intent.
    short[] getShortArrayExtra (String name)
        Retrieve extended data from the intent.
    short getShortExtra (String name, short defaultValue)
        Retrieve extended data from the intent.
    Rect getSourceBounds ()
        Get the bounds of the sender of this intent, in screen coordinates.
    String[] getStringArrayExtra (String name)
        Retrieve extended data from the intent.
    ArrayList<String> getStringArrayListExtra (String name)
        Retrieve extended data from the intent.
    String getStringExtra (String name)
        Retrieve extended data from the intent.
    String getType ()
        Retrieve any explicit MIME type included in the intent.
    boolean hasCategory (String category)
        Check if a category exists in the intent.
    boolean hasExtra (String name)
        Returns true if an extra value is associated with the given name.
    boolean hasFileDescriptors ()
        Returns true if the Intent's extras contain a parcelled file descriptor.
    static Intent makeMainActivity (ComponentName mainActivity)
        Create an intent to launch the main (root) activity of a task.
    makeMainSelectorActivity (String selectorAction, String selectorCategory)
    static Intent Make an Intent for the main activity of an application, without specifying a
        specific activity to run but giving a selector to find the activity.
    makeRestartActivityTask (ComponentName mainActivity)
    static Intent Make an Intent that can be used to re-launch an application's task in its
        base state.

```

```
static String normalizeMimeType(String type)
    Normalize a MIME data type.

    parseIntent (Resources resources, XmlPullParser parser, AttributeSet attrs)
static Intent  Parses the "intent" element (and its children) from XML and instantiates an
               Intent object.

static Intent  parseUri (String uri, int flags)
               Create an intent from a URI.

               putCharSequenceArrayListExtra (String name, ArrayList<CharSequence> value)
Intent         Add extended data to the intent.

               putExtra (String name, double[] value)
Intent         Add extended data to the intent.

               putExtra (String name, int value)
Intent         Add extended data to the intent.

               putExtra (String name, CharSequence value)
Intent         Add extended data to the intent.

               putExtra (String name, char value)
Intent         Add extended data to the intent.

               putExtra (String name, Bundle value)
Intent         Add extended data to the intent.

               putExtra (String name, Parcelable[] value)
Intent         Add extended data to the intent.

               putExtra (String name, Serializable value)
Intent         Add extended data to the intent.

               putExtra (String name, int[] value)
Intent         Add extended data to the intent.

               putExtra (String name, float value)
Intent         Add extended data to the intent.

               putExtra (String name, byte[] value)
Intent         Add extended data to the intent.

               putExtra (String name, long[] value)
Intent         Add extended data to the intent.

               putExtra (String name, Parcelable value)
Intent         Add extended data to the intent.

               putExtra (String name, float[] value)
Intent         Add extended data to the intent.

               putExtra (String name, long value)
Intent         Add extended data to the intent.

               putExtra (String name, String[] value)
Intent         Add extended data to the intent.

               putExtra (String name, boolean value)
Intent         Add extended data to the intent.

               putExtra (String name, boolean[] value)
Intent         Add extended data to the intent.

               putExtra (String name, short value)
Intent         Add extended data to the intent.

               putExtra (String name, double value)
Intent         Add extended data to the intent.

               putExtra (String name, short[] value)
Intent         Add extended data to the intent.

               putExtra (String name, String value)
Intent         Add extended data to the intent.
```

Intent	<code>putExtra (String name, byte value)</code> Add extended data to the intent.
Intent	<code>putExtra (String name, char[] value)</code> Add extended data to the intent.
Intent	<code>putExtra (String name, CharSequence[] value)</code> Add extended data to the intent.
Intent	<code>putExtras (Intent src)</code> Copy all extras in 'src' in to this intent.
Intent	<code>putExtras (Bundle extras)</code> Add a set of extended data to the intent.
Intent	<code>putIntegerArrayListExtra (String name, ArrayList&lt;Integer&gt; value)</code> Add extended data to the intent.
Intent	<code>putParcelableArrayListExtra (String name, ArrayList&lt;? extends Parcelable&gt; value)</code> Add extended data to the intent.
Intent	<code>putStringArrayListExtra (String name, ArrayList&lt;String&gt; value)</code> Add extended data to the intent.
	<code>void readFromParcel (Parcel in)</code>
	<code>void removeCategory (String category)</code> Remove a category from an intent.
	<code>void removeExtra (String name)</code> Remove extended data from the intent.
Intent	<code>replaceExtras (Bundle extras)</code> Completely replace the extras in the Intent with the given Bundle of extras.
Intent	<code>replaceExtras (Intent src)</code> Completely replace the extras in the Intent with the extras in the given Intent.
ComponentName	<code>resolveActivity (PackageManager pm)</code> Return the Activity component that should be used to handle this intent.
	<code>resolveActivityInfo (PackageManager pm, int flags)</code>
ActivityInfo	Resolve the Intent into an <code>ActivityInfo</code> describing the activity that should execute the intent.
String	<code>resolveType (ContentResolver resolver)</code> Return the MIME data type of this intent.
String	<code>resolveType (Context context)</code> Return the MIME data type of this intent.
	<code>resolveTypeIfNeeded (ContentResolver resolver)</code>
String	Return the MIME data type of this intent, only if it will be needed for intent resolution.
Intent	<code>setAction (String action)</code> Set the general action to be performed.
	<code>setClass (Context packageContext, Class&lt;?&gt; cls)</code>
Intent	Convenience for calling <code>setComponent (ComponentName)</code> with the name returned by a <code>Class</code> object.
	<code>setClassName (Context packageContext, String className)</code>
Intent	Convenience for calling <code>setComponent (ComponentName)</code> with an explicit class name.
	<code>setClassName (String packageName, String className)</code>
Intent	Convenience for calling <code>setComponent (ComponentName)</code> with an explicit application package name and class name.
	<code>setClipData (ClipData clip)</code>
void	Set a <code>ClipData</code> associated with this Intent.
Intent	<code>setComponent (ComponentName component)</code> (Usually optional) Explicitly set the component to handle the intent.

```

Intent setData(Uri data)
    Set the data this intent is operating on.
Intent setDataAndNormalize(Uri data)
    Normalize and set the data this intent is operating on.
Intent setDataAndType(Uri data, String type)
    (Usually optional) Set the data for the intent along with an explicit MIME
    data type.
Intent setDataAndTypeAndNormalize(Uri data, String type)
    (Usually optional) Normalize and set both the data Uri and an explicit MIME
    data type.
void setExtrasClassLoader(ClassLoader loader)
    Sets the ClassLoader that will be used when unmarshalling any Parcelable
    values from the extras of this Intent.
Intent setFlags(int flags)
    Set special flags controlling how this intent is handled.
Intent setPackage(String packageName)
    (Usually optional) Set an explicit application package name that limits the
    components this Intent will resolve to.
void setSelector(Intent selector)
    Set a selector for this Intent.
void setSourceBounds(Rect r)
    Set the bounds of the sender of this intent, in screen coordinates.
Intent setType(String type)
    Set an explicit MIME data type.
Intent setTypeAndNormalize(String type)
    Normalize and set an explicit MIME data type.
String toString()
    Returns a string containing a concise, human-readable description of this
    object.
String toUri()
    This method was deprecated in API level 4. Use toUri(int) instead.
String toUri(int flags)
    Convert this Intent into a String holding a URI representation of it.
void writeToParcel(Parcel out, int flags)
    Flatten this object in to a Parcel.

```

#### Inherited Methods [\[Expand\]](#)

- From class java.lang.Object
- From interface android.os.Parcelable

## Constants

---

public static final [String](#) ACTION\_AIRPLANE\_MODE\_CHANGED

Added in [API level 1](#)

Broadcast Action: The user has switched the phone into or out of Airplane Mode. One or more radios have been turned off or on. The intent will have the following extra value:

- *state* - A boolean value indicating whether Airplane Mode is on. If true, then cell radio and possibly other radios such as bluetooth or WiFi may have also been turned off

**I** This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.AIRPLANE\_MODE"

**public static final String ACTION\_ALL\_APPS**Added in [API level 1](#)

Activity Action: List all available applications

Input: Nothing.

Output: nothing.

Constant Value: "android.intent.action.ALL\_APPS"

**public static final String ACTION\_ANSWER**Added in [API level 1](#)

Activity Action: Handle an incoming phone call.

Input: nothing.

Output: nothing.

Constant Value: "android.intent.action.ANSWER"

**public static final String ACTION\_APP\_ERROR**Added in [API level 14](#)

Activity Action: The user pressed the "Report" button in the crash/ANR dialog. This intent is delivered to the package which installed the application, usually Google Play.

Input: No data is specified. The bug report is passed in using an [EXTRA\\_BUG\\_REPORT](#) ([/reference/android/content/Intent.html#EXTRA\\_BUG\\_REPORT](/reference/android/content/Intent.html#EXTRA_BUG_REPORT)) field.

Output: Nothing.

**See Also**[EXTRA\\_BUG\\_REPORT](#)

Constant Value: "android.intent.action.APP\_ERROR"

**public static final String ACTION\_ASSIST**Added in [API level 16](#)

Activity Action: Perform assist action.

Input: [EXTRA\\_ASSIST\\_PACKAGE](#) ([/reference/android/content/Intent.html#EXTRA\\_ASSIST\\_PACKAGE](/reference/android/content/Intent.html#EXTRA_ASSIST_PACKAGE)), [EXTRA\\_ASSIST\\_CONTEXT](#) ([/reference/android/content/Intent.html#EXTRA\\_ASSIST\\_CONTEXT](/reference/android/content/Intent.html#EXTRA_ASSIST_CONTEXT)), can provide additional optional contextual information about where the user was when they requested the assist. Output: nothing.

Constant Value: "android.intent.action.ASSIST"

**public static final String ACTION\_ATTACH\_DATA**Added in [API level 1](#)

Used to indicate that some piece of data should be attached to some other place. For example, image data could be attached to a contact. It is up to the recipient to decide where the data should be attached; the intent does not specify the ultimate destination.

Input: [getData\(\)](#) ([/reference/android/content/Intent.html#getData\(\)](/reference/android/content/Intent.html#getData())) is URI of data to be attached.

Output: nothing.

Constant Value: "android.intent.action.ATTACH\_DATA"

**public static final String ACTION\_BATTERY\_CHANGED**Added in [API level 1](#)

Broadcast Action: This is a *sticky broadcast* containing the charging state, level, and other information about the battery. See [BatteryManager](#) (</reference/android/os/BatteryManager.html>) for documentation on the contents of the Intent.



You can *not* receive this through components declared in manifests, only by explicitly registering for it with `Context.registerReceiver()` ([/reference/android/content/Context.html#registerReceiver\(android.content.BroadcastReceiver, android.content.IntentFilter\)](/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter))). See [ACTION\\_BATTERY\\_LOW](/reference/android/content/Intent.html#ACTION_BATTERY_LOW) ([/reference/android/content/Intent.html#ACTION\\_BATTERY\\_LOW](/reference/android/content/Intent.html#ACTION_BATTERY_LOW)), [ACTION\\_BATTERY\\_OKAY](/reference/android/content/Intent.html#ACTION_BATTERY_OKAY) ([/reference/android/content/Intent.html#ACTION\\_BATTERY\\_OKAY](/reference/android/content/Intent.html#ACTION_BATTERY_OKAY)), [ACTION\\_POWER\\_CONNECTED](/reference/android/content/Intent.html#ACTION_POWER_CONNECTED) ([/reference/android/content/Intent.html#ACTION\\_POWER\\_CONNECTED](/reference/android/content/Intent.html#ACTION_POWER_CONNECTED)), and [ACTION\\_POWER\\_DISCONNECTED](/reference/android/content/Intent.html#ACTION_POWER_DISCONNECTED) ([/reference/android/content/Intent.html#ACTION\\_POWER\\_DISCONNECTED](/reference/android/content/Intent.html#ACTION_POWER_DISCONNECTED)) for distinct battery-related broadcasts that are sent and can be received through manifest receivers.

This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.BATTERY\_CHANGED"

#### public static final [String](#) **ACTION\_BATTERY\_LOW**

Added in [API level 1](#)

Broadcast Action: Indicates low battery condition on the device. This broadcast corresponds to the "Low battery warning" system dialog.

This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.BATTERY\_LOW"

#### public static final [String](#) **ACTION\_BATTERY\_OKAY**

Added in [API level 4](#)

Broadcast Action: Indicates the battery is now okay after being low. This will be sent after [ACTION\\_BATTERY\\_LOW](/reference/android/content/Intent.html#ACTION_BATTERY_LOW) ([/reference/android/content/Intent.html#ACTION\\_BATTERY\\_LOW](/reference/android/content/Intent.html#ACTION_BATTERY_LOW)) once the battery has gone back up to an okay state.

This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.BATTERY\_OKAY"

#### public static final [String](#) **ACTION\_BOOT\_COMPLETED**

Added in [API level 1](#)

Broadcast Action: This is broadcast once, after the system has finished booting. It can be used to perform application-specific initialization, such as installing alarms. You must hold the [RECEIVE\\_BOOT\\_COMPLETED](/reference/android/Manifest.permission.html#RECEIVE_BOOT_COMPLETED) ([/reference/android/Manifest.permission.html#RECEIVE\\_BOOT\\_COMPLETED](/reference/android/Manifest.permission.html#RECEIVE_BOOT_COMPLETED)) permission in order to receive this broadcast.

This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.BOOT\_COMPLETED"

#### public static final [String](#) **ACTION\_BUG\_REPORT**

Added in [API level 1](#)

Activity Action: Show activity for reporting a bug.

Input: Nothing.

Output: Nothing.

Constant Value: "android.intent.action.BUG\_REPORT"

#### public static final [String](#) **ACTION\_CALL**

Added in [API level 1](#)

Activity Action: Perform a call to someone specified by the data.

Input: If nothing, an empty dialer is started; else `getData()` ([/reference/android/content/Intent.html#getData\(\)](/reference/android/content/Intent.html#getData())) is URI of a phone number to be dialed or a tel: URI of an explicit phone number.

Output: nothing.

Note: there will be restrictions on which applications can initiate a call; most applications should use the [ACTION\\_DIAL](/reference/android/content/Intent.html#ACTION_DIAL) ([/reference/android/content/Intent.html#ACTION\\_DIAL](/reference/android/content/Intent.html#ACTION_DIAL)).

Note: this Intent **cannot** be used to call emergency numbers. Applications can dial emergency numbers using [ACTION\\_DIAL](/reference/android/content/Intent.html#ACTION_DIAL) ([/reference/android/content/Intent.html#ACTION\\_DIAL](/reference/android/content/Intent.html#ACTION_DIAL)), however.

Constant Value: "android.intent.action.CALL"

## public static final [String](#) ACTION\_CALL\_BUTTON

Added in [API level 1](#)

Activity Action: The user pressed the "call" button to go to the dialer or other appropriate UI for placing a call.

Input: Nothing.

Output: Nothing.

Constant Value: "android.intent.action.CALL\_BUTTON"

## public static final [String](#) ACTION\_CAMERA\_BUTTON

Added in [API level 1](#)

Broadcast Action: The "Camera Button" was pressed. Includes a single extra field, [EXTRA\\_KEY\\_EVENT](/reference/android/content/Intent.html#EXTRA_KEY_EVENT) ([/reference/android/content/Intent.html#EXTRA\\_KEY\\_EVENT](/reference/android/content/Intent.html#EXTRA_KEY_EVENT)), containing the key event that caused the broadcast.

Constant Value: "android.intent.action.CAMERA\_BUTTON"

## public static final [String](#) ACTION\_CHOOSER

Added in [API level 1](#)

Activity Action: Display an activity chooser, allowing the user to pick what they want to before proceeding. This can be used as an alternative to the standard activity picker that is displayed by the system when you try to start an activity with multiple possible matches, with these differences in behavior:

- You can specify the title that will appear in the activity chooser.
- The user does not have the option to make one of the matching activities a preferred activity, and all possible activities will always be shown even if one of them is currently marked as the preferred activity.

This action should be used when the user will naturally expect to select an activity in order to proceed. An example if when not to use it is when the user clicks on a "mailto:" link. They would naturally expect to go directly to their mail app, so `startActivity()` should be called directly: it will either launch the current preferred app, or put up a dialog allowing the user to pick an app to use and optionally marking that as preferred.

In contrast, if the user is selecting a menu item to send a picture they are viewing to someone else, there are many different things they may want to do at this point: send it through e-mail, upload it to a web service, etc. In this case the CHOOSER action should be used, to always present to the user a list of the things they can do, with a nice title given by the caller such as "Send this photo with".

If you need to grant URI permissions through a chooser, you must specify the permissions to be granted on the ACTION\_CHOOSER Intent *in addition* to the EXTRA\_INTENT inside.

This means using [setClipData\(ClipData\)](/reference/android/content/Intent.html#setClipData(android.content.ClipData)) ([/reference/android/content/Intent.html#setClipData\(android.content.ClipData\)](/reference/android/content/Intent.html#setClipData(android.content.ClipData))) to specify the URIs to be granted as well as [FLAG\\_GRANT\\_READ\\_URI\\_PERMISSION](/reference/android/content/Intent.html#FLAG_GRANT_READ_URI_PERMISSION) ([/reference/android/content/Intent.html#FLAG\\_GRANT\\_READ\\_URI\\_PERMISSION](/reference/android/content/Intent.html#FLAG_GRANT_READ_URI_PERMISSION)) and/or [FLAG\\_GRANT\\_WRITE\\_URI\\_PERMISSION](/reference/android/content/Intent.html#FLAG_GRANT_WRITE_URI_PERMISSION) ([/reference/android/content/Intent.html#FLAG\\_GRANT\\_WRITE\\_URI\\_PERMISSION](/reference/android/content/Intent.html#FLAG_GRANT_WRITE_URI_PERMISSION)) as appropriate.

As a convenience, an Intent of this form can be created with the

`createChooser(Intent, CharSequence)` ([/reference/android/content/Intent.html#createChooser\(android.content.Intent, java.lang.CharSequence\)\)](/reference/android/content/Intent.html#createChooser(android.content.Intent, java.lang.CharSequence))) function.

Input: No data should be specified. `get*Extra` must have a `EXTRA_INTENT` ([/reference/android/content/Intent.html#EXTRA\\_INTENT](/reference/android/content/Intent.html#EXTRA_INTENT)) field containing the Intent being executed, and can optionally have a `EXTRA_TITLE` ([/reference/android/content/Intent.html#EXTRA\\_TITLE](/reference/android/content/Intent.html#EXTRA_TITLE)) field containing the title text to display in the chooser.

Output: Depends on the protocol of `EXTRA_INTENT` ([/reference/android/content/Intent.html#EXTRA\\_INTENT](/reference/android/content/Intent.html#EXTRA_INTENT)).

Constant Value: "android.intent.action.CHOOSER"

public static final **String** **ACTION\_CLOSE\_SYSTEM\_DIALOGS**

Added in [API level 1](#)

Broadcast Action: This is broadcast when a user action should request a temporary system dialog to dismiss. Some examples of temporary system dialogs are the notification window-shade and the recent tasks dialog.

Constant Value: "android.intent.action.CLOSE\_SYSTEM\_DIALOGS"

public static final **String** **ACTION\_CONFIGURATION\_CHANGED**

Added in [API level 1](#)

Broadcast Action: The current device [Configuration](/reference/android/content/res/Configuration.html) ([/res/Configuration.html](/reference/android/content/res/Configuration.html)) (orientation, locale, etc) has changed. When such a change happens, the UIs (view hierarchy) will need to be rebuilt based on this new information; for the most part, applications don't need to worry about this, because the system will take care of stopping and restarting the application to make sure it sees the new changes. Some system code that can not be restarted will need to watch for this action and handle it appropriately.

You can *not* receive this through components declared in manifests, only by explicitly registering for it with `Context.registerReceiver()` ([/reference/android/content/Context.html#registerReceiver\(android.content.BroadcastReceiver, android.content.IntentFilter\)\)](/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter))).

This is a protected intent that can only be sent by the system.

#### See Also

[Configuration](#)

Constant Value: "android.intent.action.CONFIGURATION\_CHANGED"

public static final **String** **ACTION\_CREATE\_DOCUMENT**

Added in [API level 19](#)

Activity Action: Allow the user to create a new document. When invoked, the system will display the various [DocumentsProvider](/reference/android/provider/DocumentsProvider.html) (</reference/android/provider/DocumentsProvider.html>) instances installed on the device, letting the user navigate through them. The returned document may be a newly created document with no content, or it may be an existing document with the requested MIME type.

Each document is represented as a content:// URI backed by a [DocumentsProvider](/reference/android/provider/DocumentsProvider.html) (</reference/android/provider/DocumentsProvider.html>), which can be opened as a stream with `openFileDescriptor(Uri, String)` ([/reference/android/content/ContentResolver.html#openFileDescriptor\(android.net.Uri, java.lang.String\)\)](/reference/android/content/ContentResolver.html#openFileDescriptor(android.net.Uri, java.lang.String))), or queried for [DocumentsContract.Document](/reference/android/provider/DocumentsContract.Document.html) (</reference/android/provider/DocumentsContract.Document.html>) metadata.

Callers must indicate the concrete MIME type of the document being created by setting `setType(String)` ([/reference/android/content/Intent.html#setType\(java.lang.String\)\)](/reference/android/content/Intent.html#setType(java.lang.String))).

This MIME type cannot be changed after the document is created.

Callers can provide an initial display name through [EXTRA\\_TITLE](#) ([//reference/android/content/Intent.html#EXTRA\\_TITLE](#)), but the user may change this value before creating the file.

Callers must include [CATEGORY\\_OPENABLE](#) ([//reference/android/content/Intent.html#CATEGORY\\_OPENABLE](#)) in the Intent so that returned URIs can be opened with [openFileDescriptor\(Uri, String\)](#) ([//reference/android/content/ContentResolver.html#openFileDescriptor\(android.net.Uri, java.lang.String\)](#)).

Output: The URI of the item that was created. This must be a content:// URI so that any receiver can access it.

#### See Also

[DocumentsContract](#)

[ACTION\\_OPEN\\_DOCUMENT](#)

[FLAG\\_GRANT\\_PERSISTABLE\\_URI\\_PERMISSION](#)

Constant Value: "android.intent.action.CREATE\_DOCUMENT"

public static final [String](#) **ACTION\_CREATE\_SHORTCUT**

Added in [API level 1](#)

Activity Action: Creates a shortcut.

Input: Nothing.

Output: An Intent representing the shortcut. The intent must contain three extras: [SHORTCUT\\_INTENT](#) (value: Intent), [SHORTCUT\\_NAME](#) (value: String), and [SHORTCUT\\_ICON](#) (value: Bitmap) or [SHORTCUT\\_ICON\\_RESOURCE](#) (value: [ShortcutIconResource](#)).

#### See Also

[EXTRA\\_SHORTCUT\\_INTENT](#)

[EXTRA\\_SHORTCUT\\_NAME](#)

[EXTRA\\_SHORTCUT\\_ICON](#)

[EXTRA\\_SHORTCUT\\_ICON\\_RESOURCE](#)

[Intent.ShortcutIconResource](#)

Constant Value: "android.intent.action.CREATE\_SHORTCUT"

public static final [String](#) **ACTION\_DATE\_CHANGED**

Added in [API level 1](#)

Broadcast Action: The date has changed.

Constant Value: "android.intent.action.DATE\_CHANGED"

public static final [String](#) **ACTION\_DEFAULT**

Added in [API level 1](#)

A synonym for [ACTION\\_VIEW](#) ([//reference/android/content/Intent.html#ACTION\\_VIEW](#)), the "standard" action that is performed on a piece of data.

Constant Value: "android.intent.action.VIEW"

public static final [String](#) **ACTION\_DELETE**

Added in [API level 1](#)

Activity Action: Delete the given data from its container.


Input: [getData\(\)](#) ([//reference/android/content/Intent.html#getData\(\)](#)) is URI of data to be deleted.

Output: nothing.

Constant Value: "android.intent.action.DELETE"

public static final String **ACTION\_DEVICE\_STORAGE\_LOW** Added in API level 1


Broadcast Action: A sticky broadcast that indicates low memory condition on the device

 This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.DEVICE\_STORAGE\_LOW"

public static final String **ACTION\_DEVICE\_STORAGE\_OK** Added in API level 1

Broadcast Action: Indicates low memory condition on the device no longer exists

 This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.DEVICE\_STORAGE\_OK"

public static final String **ACTION\_DIAL** Added in API level 1

Activity Action: Dial a number as specified by the data. This shows a UI with the number being dialed, allowing the user to explicitly initiate the call.

Input: If nothing, an empty dialer is started; else `getData()` ([/reference/android/content/Intent.html#getData\(\)](/reference/android/content/Intent.html#getData())) is URI of a phone number to be dialed or a tel: URI of an explicit phone number.

Output: nothing.

Constant Value: "android.intent.action.DIAL"

public static final String **ACTION\_DOCK\_EVENT** Added in API level 5

Broadcast Action: A sticky broadcast for changes in the physical docking state of the device.

The intent will have the following extra values:


- **EXTRA\_DOCK\_STATE** - the current dock state, indicating which dock the device is physically in.

This is intended for monitoring the current physical dock state. See [UiModeManager](/reference/android/app/UiModeManager.html) (</reference/android/app/UiModeManager.html>) for the normal API dealing with dock mode changes.

Constant Value: "android.intent.action.DOCK\_EVENT"

public static final String **ACTION\_DREAMING\_STARTED** Added in API level 17


Broadcast Action: Sent after the system starts dreaming.

 This is a protected intent that can only be sent by the system. It is only sent to registered receivers.

Constant Value: "android.intent.action.DREAMING\_STARTED"

public static final String **ACTION\_DREAMING\_STOPPED** Added in API level 17

Broadcast Action: Sent after the system stops dreaming.

 This is a protected intent that can only be sent by the system. It is only sent to registered receivers.

Constant Value: "android.intent.action.DREAMING\_STOPPED"

public static final String **ACTION\_EDIT** Added in API level 1

Activity Action: Provide explicit editable access to the given data.

Input: `getData()` ([/reference/android/content/Intent.html#getData\(\)](/reference/android/content/Intent.html#getData())) is URI of data to be edited.

Output: nothing.

Constant Value: "android.intent.action.EDIT"

### public static final String **ACTION\_EXTERNAL\_APPLICATIONS\_AVAILABLE**

Added in [API level 8](#)

Broadcast Action: Resources for a set of packages (which were previously unavailable) are currently available since the media on which they exist is available. The extra data [EXTRA\\_CHANGED\\_PACKAGE\\_LIST](/reference/android/content/Intent.html#EXTRA_CHANGED_PACKAGE_LIST) ([/reference/android/content/Intent.html#EXTRA\\_CHANGED\\_PACKAGE\\_LIST](/reference/android/content/Intent.html#EXTRA_CHANGED_PACKAGE_LIST)) contains a list of packages whose availability changed. The extra data [EXTRA\\_CHANGED\\_UID\\_LIST](/reference/android/content/Intent.html#EXTRA_CHANGED_UID_LIST) ([/reference/android/content/Intent.html#EXTRA\\_CHANGED\\_UID\\_LIST](/reference/android/content/Intent.html#EXTRA_CHANGED_UID_LIST)) contains a list of uids of packages whose availability changed. Note that the packages in this list do *not* receive this broadcast. The specified set of packages are now available on the system.

Includes the following extras:

- [EXTRA\\_CHANGED\\_PACKAGE\\_LIST](#) is the set of packages whose resources(were previously unavailable) are currently available. [EXTRA\\_CHANGED\\_UID\\_LIST](#) is the set of uids of the packages whose resources(were previously unavailable) are currently available.

**This is a protected intent that can only be sent by the system.**

Constant Value: "android.intent.action.EXTERNAL\_APPLICATIONS\_AVAILABLE"

### public static final String **ACTION\_EXTERNAL\_APPLICATIONS\_UNAVAILABLE**

Added in [API level 8](#)

Broadcast Action: Resources for a set of packages are currently unavailable since the media on which they exist is unavailable. The extra data [EXTRA\\_CHANGED\\_PACKAGE\\_LIST](/reference/android/content/Intent.html#EXTRA_CHANGED_PACKAGE_LIST) ([/reference/android/content/Intent.html#EXTRA\\_CHANGED\\_PACKAGE\\_LIST](/reference/android/content/Intent.html#EXTRA_CHANGED_PACKAGE_LIST)) contains a list of packages whose availability changed. The extra data [EXTRA\\_CHANGED\\_UID\\_LIST](/reference/android/content/Intent.html#EXTRA_CHANGED_UID_LIST) ([/reference/android/content/Intent.html#EXTRA\\_CHANGED\\_UID\\_LIST](/reference/android/content/Intent.html#EXTRA_CHANGED_UID_LIST)) contains a list of uids of packages whose availability changed. The specified set of packages can no longer be launched and are practically unavailable on the system.

Includes the following extras:

- [EXTRA\\_CHANGED\\_PACKAGE\\_LIST](#) is the set of packages whose resources are no longer available. [EXTRA\\_CHANGED\\_UID\\_LIST](#) is the set of packages whose resources are no longer available.

**This is a protected intent that can only be sent by the system.**

Constant Value: "android.intent.action.EXTERNAL\_APPLICATIONS\_UNAVAILABLE"

### public static final String **ACTION\_FACTORY\_TEST**

Added in [API level 1](#)

Activity Action: Main entry point for factory tests. Only used when the device is booting in factory test node. The implementing package must be installed in the system image.

Input: nothing

Output: nothing

Constant Value: "android.intent.action.FACTORY\_TEST"

**public static final String ACTION\_GET\_CONTENT**Added in [API level 1](#)

Activity Action: Allow the user to select a particular kind of data and return it. This is different than [ACTION\\_PICK](#) ([/reference/android/content/Intent.html#ACTION\\_PICK](#)) in that here we just say what kind of data is desired, not a URI of existing data from which the user can pick. An ACTION\_GET\_CONTENT could allow the user to create the data as it runs (for example taking a picture or recording a sound), let them browse over the web and download the desired data, etc.

There are two main ways to use this action: if you want a specific kind of data, such as a person contact, you set the MIME type to the kind of data you want and launch it with [startActivity\(Intent\)](#) ([/reference/android/content/Context.html#startActivity\(android.content.Intent\)](#)). The system will then launch the best application to select that kind of data for you.

You may also be interested in any of a set of types of content the user can pick. For example, an e-mail application that wants to allow the user to add an attachment to an e-mail message can use this action to bring up a list of all of the types of content the user can attach.

In this case, you should wrap the GET\_CONTENT intent with a chooser (through [createChooser\(Intent, CharSequence\)](#) ([/reference/android/content/Intent.html#createChooser\(android.content.Intent, java.lang.CharSequence\)](#))), which will give the proper interface for the user to pick how to send your data and allow you to specify a prompt indicating what they are doing. You will usually specify a broad MIME type (such as image/\* or \*/\*), resulting in a broad range of content types the user can select from.

When using such a broad GET\_CONTENT action, it is often desirable to only pick from data that can be represented as a stream. This is accomplished by requiring the [CATEGORY\\_OPENABLE](#) ([/reference/android/content/Intent.html#CATEGORY\\_OPENABLE](#)) in the Intent.

Callers can optionally specify [EXTRA\\_LOCAL\\_ONLY](#) ([/reference/android/content/Intent.html#EXTRA\\_LOCAL\\_ONLY](#)) to request that the launched content chooser only returns results representing data that is locally available on the device. For example, if this extra is set to true then an image picker should not show any pictures that are available from a remote server but not already on the local device (thus requiring they be downloaded when opened).

If the caller can handle multiple returned items (the user performing multiple selection), then it can specify [EXTRA\\_ALLOW\\_MULTIPLE](#) ([/reference/android/content/Intent.html#EXTRA\\_ALLOW\\_MULTIPLE](#)) to indicate this.

Input: [getType\(\)](#) ([/reference/android/content/Intent.html#getType\(\)](#)) is the desired MIME type to retrieve. Note that no URI is supplied in the intent, as there are no constraints on where the returned data originally comes from. You may also include the [CATEGORY\\_OPENABLE](#) ([/reference/android/content/Intent.html#CATEGORY\\_OPENABLE](#)) if you can only accept data that can be opened as a stream. You may use [EXTRA\\_LOCAL\\_ONLY](#) ([/reference/android/content/Intent.html#EXTRA\\_LOCAL\\_ONLY](#)) to limit content selection to local data. You may use [EXTRA\\_ALLOW\\_MULTIPLE](#) ([/reference/android/content/Intent.html#EXTRA\\_ALLOW\\_MULTIPLE](#)) to allow the user to select multiple items.

Output: The URI of the item that was picked. This must be a content: URI so that any receiver can access it.

Constant Value: "android.intent.action.GET\_CONTENT"

**public static final String ACTION\_GET\_RESTRICTION\_ENTRIES**Added in [API level 18](#)

Broadcast to a specific application to query any supported restrictions to impose on restricted users. The broadcast intent contains an extra [EXTRA\\_RESTRICTIONS\\_BUNDLE](#) ([/reference/android/content/Intent.html#EXTRA\\_RESTRICTIONS\\_BUNDLE](#)) with the currently

persisted restrictions as a Bundle of key/value pairs. The value types can be Boolean, String or String[] depending on the restriction type.

The response should contain an extra [EXTRA\\_RESTRICTIONS\\_LIST](#) ([/reference/android/content/Intent.html#EXTRA\\_RESTRICTIONS\\_LIST](#)), which is of type `ArrayList<RestrictionEntry>`. It can also contain an extra [EXTRA\\_RESTRICTIONS\\_INTENT](#) ([/reference/android/content/Intent.html#EXTRA\\_RESTRICTIONS\\_INTENT](#)), which is of type `Intent`. The activity specified by that intent will be launched for a result which must contain one of the extras [EXTRA\\_RESTRICTIONS\\_LIST](#) ([/reference/android/content/Intent.html#EXTRA\\_RESTRICTIONS\\_LIST](#)) or [EXTRA\\_RESTRICTIONS\\_BUNDLE](#) ([/reference/android/content/Intent.html#EXTRA\\_RESTRICTIONS\\_BUNDLE](#)). The keys and values of the returned restrictions will be persisted.

#### See Also

[RestrictionEntry](#)

Constant Value: "android.intent.action.GET\_RESTRICTION\_ENTRIES"

public static final [String](#) **ACTION\_GTALK\_SERVICE\_CONNECTED** Added in [API level 1](#)

Broadcast Action: A GTalk connection has been established.

Constant Value: "android.intent.action.GTALK\_CONNECTED"

public static final [String](#) **ACTION\_GTALK\_SERVICE\_DISCONNECTED** Added in [API level 1](#)

Broadcast Action: A GTalk connection has been disconnected.

Constant Value: "android.intent.action.GTALK\_DISCONNECTED"

public static final [String](#) **ACTION\_HEADSET\_PLUG** Added in [API level 1](#)

Broadcast Action: Wired Headset plugged in or unplugged.

The intent will have the following extra values:

- *state* - 0 for unplugged, 1 for plugged.
- *name* - Headset type, human readable string
- *microphone* - 1 if headset has a microphone, 0 otherwise

Constant Value: "android.intent.action.HEADSET\_PLUG"

public static final [String](#) **ACTION\_INPUT\_METHOD\_CHANGED** Added in [API level 3](#)

Broadcast Action: An input method has been changed.

Constant Value: "android.intent.action.INPUT\_METHOD\_CHANGED"

public static final [String](#) **ACTION\_INSERT** Added in [API level 1](#)

Activity Action: Insert an empty item into the given container.

Input: [getData\(\)](#) ([/reference/android/content/Intent.html#getData\(\)](#)) is URI of the directory (`vnd.android.cursor.dir/*`) in which to place the data.

Output: URI of the new data that was created.

Constant Value: "android.intent.action.INSERT"

public static final [String](#) **ACTION\_INSERT\_OR\_EDIT** Added in [API level 1](#)

Activity Action: Pick an existing item, or insert a new item, and then edit it.



Input: `getType()` ([/reference/android/content/Intent.html#getType\(\)](/reference/android/content/Intent.html#getType())) is the desired MIME type of the item to create or edit. The extras can contain type specific data to pass through to the editing/creating activity.

Output: The URI of the item that was picked. This must be a content: URI so that any receiver can access it.

Constant Value: "android.intent.action.INSERT\_OR\_EDIT"

## public static final [String](#) ACTION\_INSTALL\_PACKAGE

Added in [API level 14](#)

Activity Action: Launch application installer.

Input: The data must be a content: or file: URI at which the application can be retrieved. As of [JELLY\\_BEAN\\_MR1](#) ([/reference/android/os/Build.VERSION\\_CODES.html#JELLY\\_BEAN\\_MR1](/reference/android/os/Build.VERSION_CODES.html#JELLY_BEAN_MR1)), you can also use "package:" to install an application for the current user that is already installed for another user. You can optionally supply [EXTRA\\_INSTALLER\\_PACKAGE\\_NAME](#) ([/reference/android/content/Intent.html#EXTRA\\_INSTALLER\\_PACKAGE\\_NAME](/reference/android/content/Intent.html#EXTRA_INSTALLER_PACKAGE_NAME)), [EXTRA\\_NOT\\_UNKNOWN\\_SOURCE](#) ([/reference/android/content/Intent.html#EXTRA\\_NOT\\_UNKNOWN\\_SOURCE](/reference/android/content/Intent.html#EXTRA_NOT_UNKNOWN_SOURCE)), [EXTRA\\_ALLOW\\_REPLACE](#) ([/reference/android/content/Intent.html#EXTRA\\_ALLOW\\_REPLACE](/reference/android/content/Intent.html#EXTRA_ALLOW_REPLACE)), and [EXTRA\\_RETURN\\_RESULT](#) ([/reference/android/content/Intent.html#EXTRA\\_RETURN\\_RESULT](/reference/android/content/Intent.html#EXTRA_RETURN_RESULT)).

Output: If [EXTRA\\_RETURN\\_RESULT](#) ([/reference/android/content/Intent.html#EXTRA\\_RETURN\\_RESULT](/reference/android/content/Intent.html#EXTRA_RETURN_RESULT)), returns whether the install succeeded.

### See Also

[EXTRA\\_INSTALLER\\_PACKAGE\\_NAME](#)

[EXTRA\\_NOT\\_UNKNOWN\\_SOURCE](#)


[EXTRA\\_RETURN\\_RESULT](#)

Constant Value: "android.intent.action.INSTALL\_PACKAGE"

## public static final [String](#) ACTION\_LOCALE\_CHANGED

Added in [API level 7](#)

Broadcast Action: The current device's locale has changed.

 This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.LOCALE\_CHANGED"

## public static final [String](#) ACTION\_MAIN

Added in [API level 1](#)

Activity Action: Start as a main entry point, does not expect to receive data.

Input: nothing

Output: nothing

Constant Value: "android.intent.action.MAIN"

## public static final [String](#) ACTION\_MANAGE\_NETWORK\_USAGE

Added in [API level 14](#)

Activity Action: Show settings for managing network data usage of a specific application. Applications should define an activity that offers options to control data usage.

Constant Value: "android.intent.action.MANAGE\_NETWORK\_USAGE"

## public static final [String](#) ACTION\_MANAGE\_PACKAGE\_STORAGE

Added in [API level 1](#)

Broadcast Action: Indicates low memory condition notification acknowledged by user and package management should be started. This is triggered by the user from the [ACTION\\_DEVICE\\_STORAGE\\_LOW](#) notification.

Constant Value: "android.intent.action.MANAGE\_PACKAGE\_STORAGE"

public static final String **ACTION\_MEDIA\_BAD\_REMOVAL**

Added in [API level 1](#)

Broadcast Action: External media was removed from SD card slot, but mount point was not unmounted. The path to the mount point for the removed media is contained in the Intent.mData field.

Constant Value: "android.intent.action.MEDIA\_BAD\_REMOVAL"

public static final String **ACTION\_MEDIA\_BUTTON**

Added in [API level 1](#)

Broadcast Action: The "Media Button" was pressed. Includes a single extra field, [EXTRA\\_KEY\\_EVENT](#) ([/reference/android/content/Intent.html#EXTRA\\_KEY\\_EVENT](/reference/android/content/Intent.html#EXTRA_KEY_EVENT)), containing the key event that caused the broadcast.

Constant Value: "android.intent.action.MEDIA\_BUTTON"

public static final String **ACTION\_MEDIA\_CHECKING**

Added in [API level 3](#)

Broadcast Action: External media is present, and being disk-checked The path to the mount point for the checking media is contained in the Intent.mData field.

Constant Value: "android.intent.action.MEDIA\_CHECKING"

public static final String **ACTION\_MEDIA\_EJECT**

Added in [API level 1](#)

Broadcast Action: User has expressed the desire to remove the external storage media. Applications should close all files they have open within the mount point when they receive this intent. The path to the mount point for the media to be ejected is contained in the Intent.mData field.

Constant Value: "android.intent.action.MEDIA\_EJECT"

public static final String **ACTION\_MEDIA\_MOUNTED**

Added in [API level 1](#)

Broadcast Action: External media is present and mounted at its mount point. The path to the mount point for the mounted media is contained in the Intent.mData field. The Intent contains an extra with name "read-only" and Boolean value to indicate if the media was mounted read only.

Constant Value: "android.intent.action.MEDIA\_MOUNTED"

public static final String **ACTION\_MEDIA\_NOFS**

Added in [API level 3](#)

Broadcast Action: External media is present, but is using an incompatible fs (or is blank) The path to the mount point for the checking media is contained in the Intent.mData field.

Constant Value: "android.intent.action.MEDIA\_NOFS"

public static final String **ACTION\_MEDIA\_REMOVED**

Added in [API level 1](#)

Broadcast Action: External media has been removed. The path to the mount point for the removed media is contained in the Intent.mData field.

Constant Value: "android.intent.action.MEDIA\_REMOVED"

public static final String **ACTION\_MEDIA\_SCANNER\_FINISHED**

Added in [API level 1](#)

Broadcast Action: The media scanner has finished scanning a directory. The path to the scanned directory is contained in the Intent.mData field.

Constant Value: "android.intent.action.MEDIA\_SCANNER\_FINISHED"

public static final String **ACTION\_MEDIA\_SCANNER\_SCAN\_FILE** Added in API level 1

Broadcast Action: Request the media scanner to scan a file and add it to the media database. The path to the file is contained in the Intent.mData field.

Constant Value: "android.intent.action.MEDIA\_SCANNER\_SCAN\_FILE"

public static final String **ACTION\_MEDIA\_SCANNER\_STARTED** Added in API level 1

Broadcast Action: The media scanner has started scanning a directory. The path to the directory being scanned is contained in the Intent.mData field.

Constant Value: "android.intent.action.MEDIA\_SCANNER\_STARTED"

public static final String **ACTION\_MEDIA\_SHARED** Added in API level 1

Broadcast Action: External media is unmounted because it is being shared via USB mass storage. The path to the mount point for the shared media is contained in the Intent.mData field.

Constant Value: "android.intent.action.MEDIA\_SHARED"

public static final String **ACTION\_MEDIA\_UNMOUNTABLE** Added in API level 1

Broadcast Action: External media is present but cannot be mounted. The path to the mount point for the unmountable media is contained in the Intent.mData field.

Constant Value: "android.intent.action.MEDIA\_UNMOUNTABLE"


public static final String **ACTION\_MEDIA\_UNMOUNTED** Added in API level 1

Broadcast Action: External media is present, but not mounted at its mount point. The path to the mount point for the unmounted media is contained in the Intent.mData field.

Constant Value: "android.intent.action.MEDIA\_UNMOUNTED"

public static final String **ACTION\_MY\_PACKAGE\_REPLACED** Added in API level 12

Broadcast Action: A new version of your application has been installed over an existing one. This is only sent to the application that was replaced. It does not contain any additional data; to receive it, just use an intent filter for this action.

 This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.MY\_PACKAGE\_REPLACED"

public static final String **ACTION\_NEW\_OUTGOING\_CALL** Added in API level 1

Broadcast Action: An outgoing call is about to be placed.

The Intent will have the following extra value:

- EXTRA\_PHONE\_NUMBER - the phone number originally intended to be dialed.

Once the broadcast is finished, the resultData is used as the actual number to call. If null, no call will be placed.

It is perfectly acceptable for multiple receivers to process the outgoing call in turn: for example, a parental control application might verify that the user is authorized to place the call at that time, then a number-rewriting application might add an area code if one was not specified.

For consistency, any receiver whose purpose is to prohibit phone calls should have a priority of 0, to ensure it will see the final phone number to be dialed. Any receiver whose purpose is to rewrite phone numbers to be called should have a positive priority. Negative

priorities are reserved for the system for this broadcast; using them may cause problems.

Any `BroadcastReceiver` receiving this Intent *must not* abort the broadcast.

Emergency calls cannot be intercepted using this mechanism, and other calls cannot be modified to call emergency numbers using this mechanism.

Some apps (such as VoIP apps) may want to redirect the outgoing call to use their own service instead. Those apps should first prevent the call from being placed by setting `resultData` to `null` and then start their own app to make the call.

You must hold the `PROCESS_OUTGOING_CALLS` ([/reference/android/Manifest.permission.html#PROCESS\\_OUTGOING\\_CALLS](/reference/android/Manifest.permission.html#PROCESS_OUTGOING_CALLS)) permission to receive this Intent.

**This is a protected intent that can only be sent by the system.**

Constant Value: "android.intent.action.NEW\_OUTGOING\_CALL"

## public static final `String` ACTION\_OPEN\_DOCUMENT

Added in [API level 19](#)

Activity Action: Allow the user to select and return one or more existing documents. When invoked, the system will display the various `DocumentsProvider` (</reference/android/provider/DocumentsProvider.html>) instances installed on the device, letting the user interactively navigate through them. These documents include local media, such as photos and video, and documents provided by installed cloud storage providers.

Each document is represented as a `content://` URI backed by a `DocumentsProvider` (</reference/android/provider/DocumentsProvider.html>), which can be opened as a stream with `openFileDescriptor(Uri, String)` ([/reference/android/content/ContentResolver.html#openFileDescriptor\(android.net.Uri, java.lang.String\)](/reference/android/content/ContentResolver.html#openFileDescriptor(android.net.Uri, java.lang.String))), or queried for `DocumentsContract.Document` (</reference/android/provider/DocumentsContract.Document.html>) metadata.

All selected documents are returned to the calling application with persistable read and write permission grants. If you want to maintain access to the documents across device reboots, you need to explicitly take the persistable permissions using `takePersistableUriPermission(Uri, int)` ([/reference/android/content/ContentResolver.html#takePersistableUriPermission\(android.net.Uri, int\)](/reference/android/content/ContentResolver.html#takePersistableUriPermission(android.net.Uri, int))).

Callers can restrict document selection to a specific kind of data, such as photos, by setting one or more MIME types in `EXTRA_MIME_TYPES` ([/reference/android/content/Intent.html#EXTRA\\_MIME\\_TYPES](/reference/android/content/Intent.html#EXTRA_MIME_TYPES)).

If the caller can handle multiple returned items (the user performing multiple selection), then you can specify `EXTRA_ALLOW_MULTIPLE` ([/reference/android/content/Intent.html#EXTRA\\_ALLOW\\_MULTIPLE](/reference/android/content/Intent.html#EXTRA_ALLOW_MULTIPLE)) to indicate this.

Callers must include `CATEGORY_OPENABLE` ([/reference/android/content/Intent.html#CATEGORY\\_OPENABLE](/reference/android/content/Intent.html#CATEGORY_OPENABLE)) in the Intent so that returned URIs can be opened with `openFileDescriptor(Uri, String)` ([/reference/android/content/ContentResolver.html#openFileDescriptor\(android.net.Uri, java.lang.String\)](/reference/android/content/ContentResolver.html#openFileDescriptor(android.net.Uri, java.lang.String))).

Output: The URI of the item that was picked. This must be a `content://` URI so that any receiver can access it. If multiple documents were selected, they are returned in `getClipData()` ([/reference/android/content/Intent.html#getClipData\(\)](/reference/android/content/Intent.html#getClipData())).

### See Also

[DocumentsContract](#)

[ACTION\\_CREATE\\_DOCUMENT](#)

[FLAG\\_GRANT\\_PERSISTABLE\\_URI\\_PERMISSION](#)

Constant Value: "android.intent.action.OPEN\_DOCUMENT"

**public static final String ACTION\_PACKAGE\_ADDED**Added in [API level 1](#)

Broadcast Action: A new application package has been installed on the device. The data contains the name of the package. Note that the newly installed package does *not* receive this broadcast.

May include the following extras:

- [EXTRA\\_UID](#) containing the integer uid assigned to the new package.
- [EXTRA\\_REPLACING](#) is set to true if this is following an [ACTION\\_PACKAGE\\_REMOVED](#) broadcast for the same package.

**I** This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.PACKAGE\_ADDED"

**public static final String ACTION\_PACKAGE\_CHANGED**Added in [API level 1](#)

Broadcast Action: An existing application package has been changed (e.g. a component has been enabled or disabled). The data contains the name of the package.

- [EXTRA\\_UID](#) containing the integer uid assigned to the package.
- [EXTRA\\_CHANGED\\_COMPONENT\\_NAME\\_LIST](#) containing the class name of the changed components (or the package name itself).
- [EXTRA\\_DONT\\_KILL\\_APP](#) containing boolean field to override the default action of restarting the application.

**I** This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.PACKAGE\_CHANGED"

**public static final String ACTION\_PACKAGE\_DATA\_CLEARED**Added in [API level 3](#)

Broadcast Action: The user has cleared the data of a package. This should be preceded by [ACTION\\_PACKAGE\\_RESTARTED](#) ([/reference/android/content/Intent.html#ACTION\\_PACKAGE\\_RESTARTED](/reference/android/content/Intent.html#ACTION_PACKAGE_RESTARTED)), after which all of its persistent data is erased and this broadcast sent. Note that the cleared package does *not* receive this broadcast. The data contains the name of the package.

- [EXTRA\\_UID](#) containing the integer uid assigned to the package.

**I** This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.PACKAGE\_DATA\_CLEARED"

**public static final String ACTION\_PACKAGE\_FIRST\_LAUNCH**Added in [API level 12](#)

Broadcast Action: Sent to the installer package of an application when that application is first launched (that is the first time it is moved out of the stopped state). The data contains the name of the package.

**I** This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.PACKAGE\_FIRST\_LAUNCH"

**public static final String ACTION\_PACKAGE\_FULLY\_REMOVED**Added in [API level 14](#)

Broadcast Action: An existing application package has been completely removed from the device. The data contains the name of the package. This is like [ACTION\\_PACKAGE\\_REMOVED](#) ([/reference/android/content/Intent.html#ACTION\\_PACKAGE\\_REMOVED](/reference/android/content/Intent.html#ACTION_PACKAGE_REMOVED)), but only set when [EXTRA\\_DATA\\_REMOVED](#) ([/reference/android/content/Intent.html#EXTRA\\_DATA\\_REMOVED](/reference/android/content/Intent.html#EXTRA_DATA_REMOVED)) is true and [EXTRA\\_REPLACING](#) ([/reference/android/content/Intent.html#EXTRA\\_REPLACING](/reference/android/content/Intent.html#EXTRA_REPLACING)) is false of that broadcast.

- [EXTRA\\_UID](#) containing the integer uid previously assigned to the package.

This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.PACKAGE\_FULLY\_REMOVED"

public static final String **ACTION\_PACKAGE\_INSTALL**

Added in API level 1

This constant was deprecated in API level 14.

This constant has never been used.

Broadcast Action: Trigger the download and eventual installation of a package.

Input: `getData()` ([/reference/android/content/Intent.html#getData\(\)](/reference/android/content/Intent.html#getData())) is the URI of the package file to download.

This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.PACKAGE\_INSTALL"

public static final String **ACTION\_PACKAGE\_NEEDS\_VERIFICATION**

Added in API level 14

Broadcast Action: Sent to the system package verifier when a package needs to be verified. The data contains the package URI.

This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.PACKAGE\_NEEDS\_VERIFICATION"

public static final String **ACTION\_PACKAGE\_REMOVED**

Added in API level 1

Broadcast Action: An existing application package has been removed from the device. The data contains the name of the package. The package that is being installed does *not* receive this Intent.

- EXTRA\_UID containing the integer uid previously assigned to the package.
- EXTRA\_DATA\_REMOVED is set to true if the entire application – data and code – is being removed.
- EXTRA\_REPLACING is set to true if this will be followed by an ACTION\_PACKAGE\_ADDED broadcast for the same package.

This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.PACKAGE\_REMOVED"

public static final String **ACTION\_PACKAGE\_REPLACED**

Added in API level 3

Broadcast Action: A new version of an application package has been installed, replacing an existing version that was previously installed. The data contains the name of the package.

May include the following extras:

- EXTRA\_UID containing the integer uid assigned to the new package.

This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.PACKAGE\_REPLACED"

public static final String **ACTION\_PACKAGE\_RESTARTED**

Added in API level 1

Broadcast Action: The user has restarted a package, and all of its processes have been killed. All runtime state associated with it (processes, alarms, notifications, etc) should be removed. Note that the restarted package does *not* receive this broadcast. The data contains the name of the package.

- EXTRA\_UID containing the integer uid assigned to the package.

This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.PACKAGE\_RESTARTED"

public static final String **ACTION\_PACKAGE\_VERIFIED**

Added in API level 17

Broadcast Action: Sent to the system package verifier when a package is verified. The data contains the package URI.

This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.PACKAGE\_VERIFIED"

public static final String **ACTION\_PASTE**

Added in API level 11

Activity Action: Create a new item in the given container, initializing it from the current contents of the clipboard.

Input: `getData()` ([/reference/android/content/Intent.html#getData\(\)](#)) is URI of the directory (`vnd.android.cursor.dir/*`) in which to place the data.

Output: URI of the new data that was created.

Constant Value: "android.intent.action.PASTE"

public static final String **ACTION\_PICK**

Added in API level 1

Activity Action: Pick an item from the data, returning what was selected.

Input: `getData()` ([/reference/android/content/Intent.html#getData\(\)](#)) is URI containing a directory of data (`vnd.android.cursor.dir/*`) from which to pick an item.

Output: The URI of the item that was picked.

Constant Value: "android.intent.action.PICK"

public static final String **ACTION\_PICK\_ACTIVITY**

Added in API level 1

Activity Action: Pick an activity given an intent, returning the class selected.

Input: `get*Extra` field `EXTRA_INTENT` ([/reference/android/content/Intent.html#EXTRA\\_INTENT](#)) is an Intent used with `queryIntentActivities(Intent, int)` ([/reference/android/content/pm/PackageManager.html#queryIntentActivities\(android.content.Intent, int\)](#)) to determine the set of activities from which to pick.

Output: Class name of the activity that was selected.

Constant Value: "android.intent.action.PICK\_ACTIVITY"

public static final String **ACTION\_POWER\_CONNECTED**

Added in API level 4

Broadcast Action: External power has been connected to the device. This is intended for applications that wish to register specifically to this notification. Unlike `ACTION_BATTERY_CHANGED`, applications will be woken for this and so do not have to stay active to receive this notification. This action can be used to implement actions that wait until power is available to trigger.

This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.ACTION\_POWER\_CONNECTED"

public static final String **ACTION\_POWER\_DISCONNECTED**

Added in API level 4

Broadcast Action: External power has been removed from the device. This is intended for

applications that wish to register specifically to this notification. Unlike `ACTION_BATTERY_CHANGED`, applications will be woken for this and so do not have to stay active to receive this notification. This action can be used to implement actions that wait until power is available to trigger.

**This is a protected intent that can only be sent by the system.**

Constant Value: `"android.intent.action.ACTION_POWER_DISCONNECTED"`

`public static final String ACTION_POWER_USAGE_SUMMARY` Added in API level 4

Activity Action: Show power usage information to the user.

Input: Nothing.

Output: Nothing.

Constant Value: `"android.intent.action.POWER_USAGE_SUMMARY"`

`public static final String ACTION_PROVIDER_CHANGED` Added in API level 1

Broadcast Action: Some content providers have parts of their namespace where they publish new events or items that the user may be especially interested in. For these things, they may broadcast this action when the set of interesting items change. For example, `GmailProvider` sends this notification when the set of unread mail in the inbox changes.

The data of the intent identifies which part of which provider changed. When queried through the content resolver, the data URI will return the data set in question.

The intent will have the following extra values:

- *count* - The number of items in the data set. This is the same as the number of items in the cursor returned by querying the data URI.

This intent will be sent at boot (if the count is non-zero) and when the data set changes. It is possible for the data set to change without the count changing (for example, if a new unread message arrives in the same sync operation in which a message is archived). The phone should still ring/vibrate/etc as normal in this case.

Constant Value: `"android.intent.action.PROVIDER_CHANGED"`

`public static final String ACTION_QUICK_CLOCK` Added in API level 17

Sent when the user taps on the clock widget in the system's "quick settings" area.

Constant Value: `"android.intent.action.QUICK_CLOCK"`

`public static final String ACTION_REBOOT` Added in API level 1

Broadcast Action: Have the device reboot. This is only for use by system code.

**This is a protected intent that can only be sent by the system.**

Constant Value: `"android.intent.action.REBOOT"`

`public static final String ACTION_RUN` Added in API level 1

Activity Action: Run the data, whatever that means.

Input: ? (Note: this is currently specific to the test harness.)

Output: nothing.

Constant Value: `"android.intent.action.RUN"`

`public static final String ACTION_SCREEN_OFF`



Broadcast Action: Sent after the screen turns off.

This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.SCREEN\_OFF"

public static final [String](#) **ACTION\_SCREEN\_ON**

Added in [API level 1](#)

Broadcast Action: Sent after the screen turns on.

This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.SCREEN\_ON"

public static final [String](#) **ACTION\_SEARCH**

Added in [API level 1](#)

Activity Action: Perform a search.

Input: [getStringExtra\(SearchManager.QUERY\)](#) ([/reference/android/app/SearchManager.html#QUERY](#)) is the text to search for. If empty, simply enter your search results Activity with the search UI activated.

Output: nothing.

Constant Value: "android.intent.action.SEARCH"

public static final [String](#) **ACTION\_SEARCH\_LONG\_PRESS**

Added in [API level 3](#)

Activity Action: Start action associated with long pressing on the search key.

Input: Nothing.

Output: Nothing.

Constant Value: "android.intent.action.SEARCH\_LONG\_PRESS"

public static final [String](#) **ACTION\_SEND**

Added in [API level 1](#)

Activity Action: Deliver some data to someone else. Who the data is being delivered to is not specified; it is up to the receiver of this action to ask the user where the data should be sent.

When launching a SEND intent, you should usually wrap it in a chooser (through [createChooser\(Intent, CharSequence\)](#) ([/reference/android/content/Intent.html#createChooser\(android.content.Intent, java.lang.CharSequence\)](#))), which will give the proper interface for the user to pick how to send your data and allow you to specify a prompt indicating what they are doing.

Input: [getType\(\)](#) ([/reference/android/content/Intent.html#getType\(\)](#)) is the MIME type of the data being sent. [get\\*Extra](#) can have either a [EXTRA\\_TEXT](#) ([/reference/android/content/Intent.html#EXTRA\\_TEXT](#)) or [EXTRA\\_STREAM](#) ([/reference/android/content/Intent.html#EXTRA\\_STREAM](#)) field, containing the data to be sent. If using [EXTRA\\_TEXT](#), the MIME type should be "text/plain"; otherwise it should be the MIME type of the data in [EXTRA\\_STREAM](#). Use [/\\*](#) if the MIME type is unknown (this will only allow senders that can handle generic data streams). If using [EXTRA\\_TEXT](#) ([/reference/android/content/Intent.html#EXTRA\\_TEXT](#)), you can also optionally supply [EXTRA\\_HTML\\_TEXT](#) ([/reference/android/content/Intent.html#EXTRA\\_HTML\\_TEXT](#)) for clients to retrieve your text with HTML formatting.

As of [JELLY BEAN](#) ([/reference/android/os/Build.VERSION\\_CODES.html#JELLY\\_BEAN](#)), the data being sent can be supplied through [setClipData\(ClipData\)](#) ([/reference/android/content/Intent.html#setClipData\(android.content.ClipData\)](#)). This allows you to use [FLAG\\_GRANT\\_READ\\_URI\\_PERMISSION](#) ([/reference/android/content](#)

[/Intent.html#FLAG\\_GRANT\\_READ\\_URI\\_PERMISSION](#)) when sharing content: URIs and other advanced features of [ClipData](#) ([/reference/android/content/ClipData.html](#)). If using this approach, you still must supply the same data through the [EXTRA\\_TEXT](#) ([/reference/android/content/Intent.html#EXTRA\\_TEXT](#)) or [EXTRA\\_STREAM](#) ([/reference/android/content/Intent.html#EXTRA\\_STREAM](#)) fields described below for compatibility with old applications. If you don't set a [ClipData](#), it will be copied there for you when calling [startActivity\(Intent\)](#) ([/reference/android/content/Context.html#startActivity\(android.content.Intent\)](#)).

Optional standard extras, which may be interpreted by some recipients as appropriate, are: [EXTRA\\_EMAIL](#) ([/reference/android/content/Intent.html#EXTRA\\_EMAIL](#)), [EXTRA\\_CC](#) ([/reference/android/content/Intent.html#EXTRA\\_CC](#)), [EXTRA\\_BCC](#) ([/reference/android/content/Intent.html#EXTRA\\_BCC](#)), [EXTRA\\_SUBJECT](#) ([/reference/android/content/Intent.html#EXTRA\\_SUBJECT](#)).

Output: nothing.

Constant Value: "android.intent.action.SEND"

public static final [String](#) **ACTION\_SENDTO**

Added in [API level 1](#)

Activity Action: Send a message to someone specified by the data.

Input: [getData\(\)](#) ([/reference/android/content/Intent.html#getData\(\)](#)) is URI describing the target.

Output: nothing.

Constant Value: "android.intent.action.SENDTO"

public static final [String](#) **ACTION\_SEND\_MULTIPLE**

Added in [API level 4](#)

Activity Action: Deliver multiple data to someone else.

Like [ACTION\\_SEND](#) ([/reference/android/content/Intent.html#ACTION\\_SEND](#)), except the data is multiple.

Input: [getType\(\)](#) ([/reference/android/content/Intent.html#getType\(\)](#)) is the MIME type of the data being sent. [get\\*ArrayListExtra](#) can have either a [EXTRA\\_TEXT](#) ([/reference/android/content/Intent.html#EXTRA\\_TEXT](#)) or [EXTRA\\_STREAM](#) ([/reference/android/content/Intent.html#EXTRA\\_STREAM](#)) field, containing the data to be sent. If using [EXTRA\\_TEXT](#) ([/reference/android/content/Intent.html#EXTRA\\_TEXT](#)), you can also optionally supply [EXTRA\\_HTML\\_TEXT](#) ([/reference/android/content/Intent.html#EXTRA\\_HTML\\_TEXT](#)) for clients to retrieve your text with HTML formatting.

Multiple types are supported, and receivers should handle mixed types whenever possible. The right way for the receiver to check them is to use the content resolver on each URI. The intent sender should try to put the most concrete mime type in the intent type, but it can fall back to <type>/\* or \*/\* as needed.

e.g. if you are sending image/jpg and image/png, the intent's type can be image/jpg, but if you are sending image/jpg and image/png, then the intent's type should be image/\*.

As of [JELLY\\_BEAN](#) ([/reference/android/os/Build.VERSION\\_CODES.html#JELLY\\_BEAN](#)), the data being sent can be supplied through [setClipData\(ClipData\)](#) ([/reference/android/content/Intent.html#setClipData\(android.content.ClipData\)](#)). This allows you to use [FLAG\\_GRANT\\_READ\\_URI\\_PERMISSION](#) ([/reference/android/content/Intent.html#FLAG\\_GRANT\\_READ\\_URI\\_PERMISSION](#)) when sharing content: URIs and other advanced features of [ClipData](#) ([/reference/android/content/ClipData.html](#)). If using this approach, you still must supply the same data through the [EXTRA\\_TEXT](#) ([/reference/android/content/Intent.html#EXTRA\\_TEXT](#)) or [EXTRA\\_STREAM](#) ([/reference/android/content](#)

[/Intent.html#EXTRA\\_STREAM](#)) fields described below for compatibility with old applications. If you don't set a ClipData, it will be copied there for you when calling [startActivity\(Intent\)](#) ([/reference/android/content/Context.html#startActivity\(android.content.Intent\)](#)).

Optional standard extras, which may be interpreted by some recipients as appropriate, are: [EXTRA\\_EMAIL](#) ([/reference/android/content/Intent.html#EXTRA\\_EMAIL](#)), [EXTRA\\_CC](#) ([/reference/android/content/Intent.html#EXTRA\\_CC](#)), [EXTRA\\_BCC](#) ([/reference/android/content/Intent.html#EXTRA\\_BCC](#)), [EXTRA\\_SUBJECT](#) ([/reference/android/content/Intent.html#EXTRA\\_SUBJECT](#)).

Output: nothing.

Constant Value: "android.intent.action.SEND\_MULTIPLE"

public static final [String](#) **ACTION\_SET\_WALLPAPER**

Added in [API level 1](#)

Activity Action: Show settings for choosing wallpaper

Input: Nothing.


Output: Nothing.

Constant Value: "android.intent.action.SET\_WALLPAPER"

public static final [String](#) **ACTION\_SHUTDOWN**

Added in [API level 4](#)

Broadcast Action: Device is shutting down. This is broadcast when the device is being shut down (completely turned off, not sleeping). Once the broadcast is complete, the final shutdown will proceed and all unsaved data lost. Apps will not normally need to handle this, since the foreground activity will be paused as well.

 This is a protected intent that can only be sent by the system.

May include the following extras:

- [EXTRA\\_SHUTDOWN\\_USERSPACE\\_ONLY](#) a boolean that is set to true if this shutdown is only for userspace processes. If not set, assumed to be false.

Constant Value: "android.intent.action.ACTION\_SHUTDOWN"

public static final [String](#) **ACTION\_SYNC**

Added in [API level 1](#)

Activity Action: Perform a data synchronization.

Input: ?

Output: ?

Constant Value: "android.intent.action.SYNC"

public static final [String](#) **ACTION\_SYSTEM\_TUTORIAL**

Added in [API level 3](#)

Activity Action: Start the platform-defined tutorial

Input: [getStringExtra\(SearchManager.QUERY\)](#) ([/reference/android/app/SearchManager.html#QUERY](#)) is the text to search for. If empty, simply enter your search results Activity with the search UI activated.

Output: nothing.

Constant Value: "android.intent.action.SYSTEM\_TUTORIAL"

public static final [String](#) **ACTION\_TIMEZONE\_CHANGED**

Broadcast Action: The timezone has changed. The intent will have the following extra values:

- *time-zone* - The `java.util.TimeZone.getID()` value identifying the new time zone.
- This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.TIMEZONE\_CHANGED"

public static final String **ACTION\_TIME\_CHANGED**

Added in API level 1

Broadcast Action: The time was set.

Constant Value: "android.intent.action.TIME\_SET"

public static final String **ACTION\_TIME\_TICK**

Added in API level 1

Broadcast Action: The current time has changed. Sent every minute. You can *not* receive this through components declared in manifests, only by explicitly registering for it with `Context.registerReceiver()` ([/reference/android/content/Context.html#registerReceiver\(android.content.BroadcastReceiver, android.content.IntentFilter\)](/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter))).

This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.TIME\_TICK"

public static final String **ACTION\_UID\_REMOVED**

Added in API level 1

Broadcast Action: A user ID has been removed from the system. The user ID number is stored in the extra data under `EXTRA_UID` ([/reference/android/content/Intent.html#EXTRA\\_UID](/reference/android/content/Intent.html#EXTRA_UID)).

This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.UID\_REMOVED"

public static final String **ACTION\_UMS\_CONNECTED**

Added in API level 1

This constant was deprecated in API level 14.  
replaced by `android.os.storage.StorageEventListener`

Broadcast Action: The device has entered USB Mass Storage mode. This is used mainly for the USB Settings panel. Apps should listen for `ACTION_MEDIA_MOUNTED` and `ACTION_MEDIA_UNMOUNTED` broadcasts to be notified when the SD card file system is mounted or unmounted

Constant Value: "android.intent.action.UMS\_CONNECTED"

public static final String **ACTION\_UMS\_DISCONNECTED**

Added in API level 1

This constant was deprecated in API level 14.  
replaced by `android.os.storage.StorageEventListener`

Broadcast Action: The device has exited USB Mass Storage mode. This is used mainly for the USB Settings panel. Apps should listen for `ACTION_MEDIA_MOUNTED` and `ACTION_MEDIA_UNMOUNTED` broadcasts to be notified when the SD card file system is mounted or unmounted

Constant Value: "android.intent.action.UMS\_DISCONNECTED"

public static final String **ACTION\_UNINSTALL\_PACKAGE**

Added in API level 14

Activity Action: Launch application uninstaller.

Input: The data must be a package: URI whose scheme specific part is the package name of the current installed package to be uninstalled. You can optionally supply [EXTRA\\_RETURN\\_RESULT](/reference/android/content/Intent.html#EXTRA_RETURN_RESULT) ([/reference/android/content/Intent.html#EXTRA\\_RETURN\\_RESULT](/reference/android/content/Intent.html#EXTRA_RETURN_RESULT)).

Output: If [EXTRA\\_RETURN\\_RESULT](/reference/android/content/Intent.html#EXTRA_RETURN_RESULT) ([/reference/android/content/Intent.html#EXTRA\\_RETURN\\_RESULT](/reference/android/content/Intent.html#EXTRA_RETURN_RESULT)), returns whether the install succeeded.

Constant Value: "android.intent.action.UNINSTALL\_PACKAGE"

#### public static final [String](#) **ACTION\_USER\_BACKGROUND**

Added in [API level 17](#)

Sent when a user switch is happening, causing the process's user to be sent to the background. This is only sent to receivers registered through [Context.registerReceiver](/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter)) ([/reference/android/content/Context.html#registerReceiver\(android.content.BroadcastReceiver, android.content.IntentFilter\)](/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter))). It is sent to the user that is going to the background. This is sent as a foreground broadcast, since it is part of a visible user interaction; be as quick as possible when handling it.

Constant Value: "android.intent.action.USER\_BACKGROUND"

#### public static final [String](#) **ACTION\_USER\_FOREGROUND**

Added in [API level 17](#)

Sent when a user switch is happening, causing the process's user to be brought to the foreground. This is only sent to receivers registered through [Context.registerReceiver](/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter)) ([/reference/android/content/Context.html#registerReceiver\(android.content.BroadcastReceiver, android.content.IntentFilter\)](/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver, android.content.IntentFilter))). It is sent to the user that is going to the foreground. This is sent as a foreground broadcast, since it is part of a visible user interaction; be as quick as possible when handling it.

Constant Value: "android.intent.action.USER\_FOREGROUND"

#### public static final [String](#) **ACTION\_USER\_INITIALIZE**

Added in [API level 17](#)


Sent the first time a user is starting, to allow system apps to perform one time initialization. (This will not be seen by third party applications because a newly initialized user does not have any third party applications installed for it.) This is sent early in starting the user, around the time the home app is started, before [ACTION\\_BOOT\\_COMPLETED](/reference/android/content/Intent.html#ACTION_BOOT_COMPLETED) ([/reference/android/content/Intent.html#ACTION\\_BOOT\\_COMPLETED](/reference/android/content/Intent.html#ACTION_BOOT_COMPLETED)) is sent. This is sent as a foreground broadcast, since it is part of a visible user interaction; be as quick as possible when handling it.

Constant Value: "android.intent.action.USER\_INITIALIZE"

#### public static final [String](#) **ACTION\_USER\_PRESENT**

Added in [API level 3](#)

Broadcast Action: Sent when the user is present after device wakes up (e.g when the keyguard is gone).

 This is a protected intent that can only be sent by the system.

Constant Value: "android.intent.action.USER\_PRESENT"

#### public static final [String](#) **ACTION\_VIEW**

Added in [API level 1](#)

Activity Action: Display the data to the user. This is the most common action performed on data – it is the generic action you can use on a piece of data to get the most reasonable thing to occur. For example, when used on a contacts entry it will view the entry; when used on a mailto: URI it will bring up a compose window filled with the information supplied by the URI; when used with a tel: URI it will invoke the dialer.

Input: `getData()` ([/reference/android/content/Intent.html#getData\(\)](/reference/android/content/Intent.html#getData())) is URI from which to retrieve data.

Output: nothing.

Constant Value: "android.intent.action.VIEW"

public static final **String ACTION\_VOICE\_COMMAND**

Added in [API level 1](#)

Activity Action: Start Voice Command.

Input: Nothing.

Output: Nothing.

Constant Value: "android.intent.action.VOICE\_COMMAND"

public static final **String ACTION\_WALLPAPER\_CHANGED**

Added in [API level 1](#)

This constant was deprecated in API level 16.

Modern applications should use

[WindowManager.LayoutParams.FLAG\\_SHOW\\_WALLPAPER](#) ([/reference/android/view/WindowManager.LayoutParams.html#FLAG\\_SHOW\\_WALLPAPER](/reference/android/view/WindowManager.LayoutParams.html#FLAG_SHOW_WALLPAPER)) to have the wallpaper shown behind their UI, rather than watching for this broadcast and rendering the wallpaper on their own.

Broadcast Action: The current system wallpaper has changed. See [WallpaperManager](#) (</reference/android/app/WallpaperManager.html>) for retrieving the new wallpaper. This should *only* be used to determine when the wallpaper has changed to show the new wallpaper to the user. You should certainly never, in response to this, change the wallpaper or other attributes of it such as the suggested size. That would be crazy, right? You'd cause all kinds of loops, especially if other apps are doing similar things, right? Of course. So please don't do this.

Constant Value: "android.intent.action.WALLPAPER\_CHANGED"

public static final **String ACTION\_WEB\_SEARCH**

Added in [API level 1](#)

Activity Action: Perform a web search.

Input: `getStringExtra(SearchManager.QUERY)` (</reference/android/app/SearchManager.html#QUERY>) is the text to search for. If it is a url starts with http or https, the site will be opened. If it is plain text, Google search will be applied.

Output: nothing.

Constant Value: "android.intent.action.WEB\_SEARCH"

public static final **String CATEGORY\_ALTERNATIVE**

Added in [API level 1](#)

Set if the activity should be considered as an alternative action to the data the user is currently viewing. See also [CATEGORY\\_SELECTED\\_ALTERNATIVE](#) ([/reference/android/content/Intent.html#CATEGORY\\_SELECTED\\_ALTERNATIVE](/reference/android/content/Intent.html#CATEGORY_SELECTED_ALTERNATIVE)) for an alternative action that applies to the selection in a list of items.

Supporting this category means that you would like your activity to be displayed in the set of alternative things the user can do, usually as part of the current activity's options menu. You will usually want to include a specific label in the `<intent-filter>` of this action describing to the user what it does.

The action of `IntentFilter` with this category is important in that it describes the specific action the target will perform. This generally should not be a generic action (such as [ACTION\\_VIEW](#) ([/reference/android/content/Intent.html#ACTION\\_VIEW](/reference/android/content/Intent.html#ACTION_VIEW)), but rather a specific

name such as "com.android.camera.action.CROP. Only one alternative of any particular action will be shown to the user, so using a specific action like this makes sure that your alternative will be displayed while also allowing other applications to provide their own overrides of that particular action.

Constant Value: "android.intent.category.ALTERNATIVE"

#### public static final String **CATEGORY\_APP\_BROWSER**

Added in [API level 15](#)

Used with [ACTION\\_MAIN](#) ([/reference/android/content/Intent.html#ACTION\\_MAIN](#)) to launch the browser application. The activity should be able to browse the Internet.

NOTE: This should not be used as the primary key of an Intent, since it will not result in the app launching with the correct action and category. Instead, use this with [makeMainSelectorActivity\(String, String\)](#) ([/reference/android/content/Intent.html#makeMainSelectorActivity\(java.lang.String, java.lang.String\)](#)) to generate a main Intent with this category in the selector.

Constant Value: "android.intent.category.APP\_BROWSER"

#### public static final String **CATEGORY\_APP\_CALCULATOR**

Added in [API level 15](#)

Used with [ACTION\\_MAIN](#) ([/reference/android/content/Intent.html#ACTION\\_MAIN](#)) to launch the calculator application. The activity should be able to perform standard arithmetic operations.

NOTE: This should not be used as the primary key of an Intent, since it will not result in the app launching with the correct action and category. Instead, use this with [makeMainSelectorActivity\(String, String\)](#) ([/reference/android/content/Intent.html#makeMainSelectorActivity\(java.lang.String, java.lang.String\)](#)) to generate a main Intent with this category in the selector.

Constant Value: "android.intent.category.APP\_CALCULATOR"

#### public static final String **CATEGORY\_APP\_CALENDAR**

Added in [API level 15](#)

Used with [ACTION\\_MAIN](#) ([/reference/android/content/Intent.html#ACTION\\_MAIN](#)) to launch the calendar application. The activity should be able to view and manipulate calendar entries.

NOTE: This should not be used as the primary key of an Intent, since it will not result in the app launching with the correct action and category. Instead, use this with [makeMainSelectorActivity\(String, String\)](#) ([/reference/android/content/Intent.html#makeMainSelectorActivity\(java.lang.String, java.lang.String\)](#)) to generate a main Intent with this category in the selector.

Constant Value: "android.intent.category.APP\_CALENDAR"

#### public static final String **CATEGORY\_APP\_CONTACTS**

Added in [API level 15](#)

Used with [ACTION\\_MAIN](#) ([/reference/android/content/Intent.html#ACTION\\_MAIN](#)) to launch the contacts application. The activity should be able to view and manipulate address book entries.

NOTE: This should not be used as the primary key of an Intent, since it will not result in the app launching with the correct action and category. Instead, use this with [makeMainSelectorActivity\(String, String\)](#) ([/reference/android/content/Intent.html#makeMainSelectorActivity\(java.lang.String, java.lang.String\)](#)) to generate a main Intent with this category in the selector.

Constant Value: "android.intent.category.APP\_CONTACTS"

#### public static final String **CATEGORY\_APP\_EMAIL**

Used with [ACTION\\_MAIN](/reference/android/content/Intent.html#ACTION_MAIN) (/reference/android/content/Intent.html#ACTION\_MAIN) to launch the email application. The activity should be able to send and receive email.

NOTE: This should not be used as the primary key of an Intent, since it will not result in the app launching with the correct action and category. Instead, use this with [makeMainSelectorActivity\(String, String\)](/reference/android/content/Intent.html#makeMainSelectorActivity(java.lang.String, java.lang.String)) (/reference/android/content/Intent.html#makeMainSelectorActivity(java.lang.String, java.lang.String)) to generate a main Intent with this category in the selector.

Constant Value: "android.intent.category.APP\_EMAIL"

public static final [String](#) **CATEGORY\_APP\_GALLERY**

Added in [API level 15](#)

Used with [ACTION\\_MAIN](/reference/android/content/Intent.html#ACTION_MAIN) (/reference/android/content/Intent.html#ACTION\_MAIN) to launch the gallery application. The activity should be able to view and manipulate image and video files stored on the device.

NOTE: This should not be used as the primary key of an Intent, since it will not result in the app launching with the correct action and category. Instead, use this with [makeMainSelectorActivity\(String, String\)](/reference/android/content/Intent.html#makeMainSelectorActivity(java.lang.String, java.lang.String)) (/reference/android/content/Intent.html#makeMainSelectorActivity(java.lang.String, java.lang.String)) to generate a main Intent with this category in the selector.

Constant Value: "android.intent.category.APP\_GALLERY"

public static final [String](#) **CATEGORY\_APP\_MAPS**

Added in [API level 15](#)

Used with [ACTION\\_MAIN](/reference/android/content/Intent.html#ACTION_MAIN) (/reference/android/content/Intent.html#ACTION\_MAIN) to launch the maps application. The activity should be able to show the user's current location and surroundings.

NOTE: This should not be used as the primary key of an Intent, since it will not result in the app launching with the correct action and category. Instead, use this with [makeMainSelectorActivity\(String, String\)](/reference/android/content/Intent.html#makeMainSelectorActivity(java.lang.String, java.lang.String)) (/reference/android/content/Intent.html#makeMainSelectorActivity(java.lang.String, java.lang.String)) to generate a main Intent with this category in the selector.

Constant Value: "android.intent.category.APP\_MAPS"

public static final [String](#) **CATEGORY\_APP\_MARKET**

Added in [API level 11](#)

This activity allows the user to browse and download new applications.

Constant Value: "android.intent.category.APP\_MARKET"

public static final [String](#) **CATEGORY\_APP\_MESSAGING**

Added in [API level 15](#)

Used with [ACTION\\_MAIN](/reference/android/content/Intent.html#ACTION_MAIN) (/reference/android/content/Intent.html#ACTION\_MAIN) to launch the messaging application. The activity should be able to send and receive text messages.

NOTE: This should not be used as the primary key of an Intent, since it will not result in the app launching with the correct action and category. Instead, use this with [makeMainSelectorActivity\(String, String\)](/reference/android/content/Intent.html#makeMainSelectorActivity(java.lang.String, java.lang.String)) (/reference/android/content/Intent.html#makeMainSelectorActivity(java.lang.String, java.lang.String)) to generate a main Intent with this category in the selector.

Constant Value: "android.intent.category.APP\_MESSAGING"

public static final [String](#) **CATEGORY\_APP\_MUSIC**

Added in [API level 15](#)

Used with [ACTION\\_MAIN](/reference/android/content/Intent.html#ACTION_MAIN) (/reference/android/content/Intent.html#ACTION\_MAIN) to launch the music application. The activity should be able to play, browse, or manipulate music files



stored on the device.

NOTE: This should not be used as the primary key of an Intent, since it will not result in the app launching with the correct action and category. Instead, use this with [makeMainSelectorActivity\(String, String\)](#) ([/reference/android/content/Intent.html#makeMainSelectorActivity\(java.lang.String, java.lang.String\)](#)) to generate a main Intent with this category in the selector.

Constant Value: "android.intent.category.APP\_MUSIC"

#### public static final [String](#) **CATEGORY\_BROWSABLE**

Added in [API level 1](#)

Activities that can be safely invoked from a browser must support this category. For example, if the user is viewing a web page or an e-mail and clicks on a link in the text, the Intent generated execute that link will require the BROWSABLE category, so that only activities supporting this category will be considered as possible actions. By supporting this category, you are promising that there is nothing damaging (without user intervention) that can happen by invoking any matching Intent.

Constant Value: "android.intent.category.BROWSABLE"

#### public static final [String](#) **CATEGORY\_CAR\_DOCK**

Added in [API level 5](#)

An activity to run when device is inserted into a car dock. Used with [ACTION\\_MAIN](#) ([/reference/android/content/Intent.html#ACTION\\_MAIN](#)) to launch an activity. For more information, see [UiModeManager](#) ([/reference/android/app/UiModeManager.html](#)).

Constant Value: "android.intent.category.CAR\_DOCK"

#### public static final [String](#) **CATEGORY\_CAR\_MODE**

Added in [API level 8](#)

Used to indicate that the activity can be used in a car environment.

Constant Value: "android.intent.category.CAR\_MODE"

#### public static final [String](#) **CATEGORY\_DEFAULT**

Added in [API level 1](#)

Set if the activity should be an option for the default action (center press) to perform on a piece of data. Setting this will hide from the user any activities without it set when performing an action on some data. Note that this is normally -not- set in the Intent when initiating an action -- it is for use in intent filters specified in packages.

Constant Value: "android.intent.category.DEFAULT"

#### public static final [String](#) **CATEGORY\_DESK\_DOCK**

Added in [API level 5](#)

An activity to run when device is inserted into a car dock. Used with [ACTION\\_MAIN](#) ([/reference/android/content/Intent.html#ACTION\\_MAIN](#)) to launch an activity. For more information, see [UiModeManager](#) ([/reference/android/app/UiModeManager.html](#)).

Constant Value: "android.intent.category.DESK\_DOCK"

#### public static final [String](#) **CATEGORY\_DEVELOPMENT\_PREFERENCE**

Added in [API level 1](#)

This activity is a development preference panel.

Constant Value: "android.intent.category.DEVELOPMENT\_PREFERENCE"

#### public static final [String](#) **CATEGORY\_EMBED**

Added in [API level 1](#)

Capable of running inside a parent activity container.

Constant Value: "android.intent.category.EMBED"

public static final [String](#) **CATEGORY\_FRAMEWORK\_INSTRUMENTATION\_TEST** Added in [API level 1](#)

To be used as code under test for framework instrumentation tests.

Constant Value: "android.intent.category.FRAMEWORK\_INSTRUMENTATION\_TEST"

public static final [String](#) **CATEGORY\_HE\_DESK\_DOCK** Added in [API level 11](#)

An activity to run when device is inserted into a digital (high end) dock. Used with [ACTION\\_MAIN](#) ([/reference/android/content/Intent.html#ACTION\\_MAIN](#)) to launch an activity. For more information, see [UiModeManager](#) ([/reference/android/app/UiModeManager.html](#)).

Constant Value: "android.intent.category.HE\_DESK\_DOCK"

public static final [String](#) **CATEGORY\_HOME** Added in [API level 1](#)

This is the home activity, that is the first activity that is displayed when the device boots.

Constant Value: "android.intent.category.HOME"

public static final [String](#) **CATEGORY\_INFO** Added in [API level 3](#)

Provides information about the package it is in; typically used if a package does not contain a [CATEGORY\\_LAUNCHER](#) ([/reference/android/content/Intent.html#CATEGORY\\_LAUNCHER](#)) to provide a front-door to the user without having to be shown in the all apps list.

Constant Value: "android.intent.category.INFO"

public static final [String](#) **CATEGORY\_LAUNCHER** Added in [API level 1](#)

Should be displayed in the top-level launcher.

Constant Value: "android.intent.category.LAUNCHER"

public static final [String](#) **CATEGORY\_LE\_DESK\_DOCK** Added in [API level 11](#)

An activity to run when device is inserted into a analog (low end) dock. Used with [ACTION\\_MAIN](#) ([/reference/android/content/Intent.html#ACTION\\_MAIN](#)) to launch an activity. For more information, see [UiModeManager](#) ([/reference/android/app/UiModeManager.html](#)).

Constant Value: "android.intent.category.LE\_DESK\_DOCK"

public static final [String](#) **CATEGORY\_MONKEY** Added in [API level 1](#)

This activity may be exercised by the monkey or other automated test tools.

Constant Value: "android.intent.category.MONKEY"

public static final [String](#) **CATEGORY\_OPENABLE** Added in [API level 1](#)

Used to indicate that an intent only wants URIs that can be opened with [openFileDescriptor\(Uri, String\)](#) ([/reference/android/content/ContentResolver.html#openFileDescriptor\(android.net.Uri, java.lang.String\)](#)). Openable URIs must support at least the columns defined in [OpenableColumns](#) ([/reference/android/provider/OpenableColumns.html](#)) when queried.

#### See Also

[ACTION\\_GET\\_CONTENT](#)

[ACTION\\_OPEN\\_DOCUMENT](#)

[ACTION\\_CREATE\\_DOCUMENT](#)

Constant Value: "android.intent.category.OPENABLE"

public static final String **CATEGORY\_PREFERENCE**

Added in API level 1

This activity is a preference panel.

Constant Value: "android.intent.category.PREFERENCE"

public static final String **CATEGORY\_SAMPLE\_CODE**

Added in API level 1

To be used as a sample code example (not part of the normal user experience).

Constant Value: "android.intent.category.SAMPLE\_CODE"

public static final String **CATEGORY\_SELECTED\_ALTERNATIVE**

Added in API level 1

Set if the activity should be considered as an alternative selection action to the data the user has currently selected. This is like CATEGORY\_ALTERNATIVE ([/reference/android/content/Intent.html#CATEGORY\\_ALTERNATIVE](/reference/android/content/Intent.html#CATEGORY_ALTERNATIVE)), but is used in activities showing a list of items from which the user can select, giving them alternatives to the default action that will be performed on it.

Constant Value: "android.intent.category.SELECTED\_ALTERNATIVE"

public static final String **CATEGORY\_TAB**

Added in API level 1

Intended to be used as a tab inside of a containing TabActivity.

Constant Value: "android.intent.category.TAB"

public static final String **CATEGORY\_TEST**

Added in API level 1

To be used as a test (not part of the normal user experience).

Constant Value: "android.intent.category.TEST"

public static final String **CATEGORY\_UNIT\_TEST**

Added in API level 1

To be used as a unit test (run through the Test Harness).

Constant Value: "android.intent.category.UNIT\_TEST"

public static final String **EXTRA\_ALARM\_COUNT**

Added in API level 1

Used as an int extra field in AlarmManager (</reference/android/app/AlarmManager.html>) intents to tell the application being invoked how many pending alarms are being delivered with the intent. For one-shot alarms this will always be 1. For recurring alarms, this might be greater than 1 if the device was asleep or powered off at the time an earlier alarm would have been delivered.

Constant Value: "android.intent.extra.ALARM\_COUNT"

public static final String **EXTRA\_ALLOW\_MULTIPLE**

Added in API level 18

Extra used to indicate that an intent can allow the user to select and return multiple items. This is a boolean extra; the default is false. If true, an implementation is allowed to present the user with a UI where they can pick multiple items that are all returned to the caller. When this happens, they should be returned as the getClipData() ([/reference/android/content/Intent.html#getClipData\(\)](/reference/android/content/Intent.html#getClipData())) part of the result Intent.

#### See Also

[ACTION\\_GET\\_CONTENT](#)

[ACTION\\_OPEN\\_DOCUMENT](#)

Constant Value: "android.intent.extra.ALLOW\_MULTIPLE"

**public static final String EXTRA\_ALLOW\_REPLACE**Added in [API level 14](#)

This constant was deprecated in API level 16.

As of [JELLY\\_BEAN](#) ([/reference/android/os/Build.VERSION\\_CODES.html#JELLY\\_BEAN](#)), Android will no longer show an interstitial message about updating existing applications so this is no longer needed.

Used as a boolean extra field with [ACTION\\_INSTALL\\_PACKAGE](#) ([/reference/android/content/Intent.html#ACTION\\_INSTALL\\_PACKAGE](#)) to install a package. Tells the installer UI to skip the confirmation with the user if the .apk is replacing an existing one.

Constant Value: "android.intent.extra.ALLOW\_REPLACE"

**public static final String EXTRA\_ASSIST\_CONTEXT**Added in [API level 18](#)

An optional field on [ACTION\\_ASSIST](#) ([/reference/android/content/Intent.html#ACTION\\_ASSIST](#)) and containing additional contextual information supplied by the current foreground app at the time of the assist request. This is a [Bundle](#) ([/reference/android/os/Bundle.html](#)) of additional data.

Constant Value: "android.intent.extra.ASSIST\_CONTEXT"

**public static final String EXTRA\_ASSIST\_PACKAGE**Added in [API level 18](#)

An optional field on [ACTION\\_ASSIST](#) ([/reference/android/content/Intent.html#ACTION\\_ASSIST](#)) containing the name of the current foreground application package at the time the assist was invoked.

Constant Value: "android.intent.extra.ASSIST\_PACKAGE"

**public static final String EXTRA\_BCC**Added in [API level 1](#)

A `String[]` holding e-mail addresses that should be blind carbon copied.

Constant Value: "android.intent.extra.BCC"

**public static final String EXTRA\_BUG\_REPORT**Added in [API level 14](#)

Used as a parcelable extra field in [ACTION\\_APP\\_ERROR](#) ([/reference/android/content/Intent.html#ACTION\\_APP\\_ERROR](#)), containing the bug report.

Constant Value: "android.intent.extra.BUG\_REPORT"

**public static final String EXTRA\_CC**Added in [API level 1](#)

A `String[]` holding e-mail addresses that should be carbon copied.

Constant Value: "android.intent.extra.CC"

**public static final String EXTRA\_CHANGED\_COMPONENT\_NAME**Added in [API level 5](#)

This constant was deprecated in API level 7.

See [EXTRA\\_CHANGED\\_COMPONENT\\_NAME\\_LIST](#) ([/reference/android/content/Intent.html#EXTRA\\_CHANGED\\_COMPONENT\\_NAME\\_LIST](#)); this field will contain only the first name in the list.

Constant Value: "android.intent.extra.changed\_component\_name"

**public static final String EXTRA\_CHANGED\_COMPONENT\_NAME\_LIST**Added in [API level 7](#)

This field is part of [ACTION\\_PACKAGE\\_CHANGED](#) ([/reference/android/content](#)

[/Intent.html#ACTION\\_PACKAGE\\_CHANGED](#)), and contains a string array of all of the components that have changed. If the state of the overall package has changed, then it will contain an entry with the package name itself.

Constant Value: "android.intent.extra.changed\_component\_name\_list"

public static final **String** **EXTRA\_CHANGED\_PACKAGE\_LIST** Added in [API level 8](#)

This field is part of [ACTION\\_EXTERNAL\\_APPLICATIONS\\_AVAILABLE](#) ([/reference/android/content/Intent.html#ACTION\\_EXTERNAL\\_APPLICATIONS\\_AVAILABLE](#)), [ACTION\\_EXTERNAL\\_APPLICATIONS\\_UNAVAILABLE](#) ([/reference/android/content/Intent.html#ACTION\\_EXTERNAL\\_APPLICATIONS\\_UNAVAILABLE](#)) and contains a string array of all of the components that have changed.

Constant Value: "android.intent.extra.changed\_package\_list"

public static final **String** **EXTRA\_CHANGED\_UID\_LIST** Added in [API level 8](#)

This field is part of [ACTION\\_EXTERNAL\\_APPLICATIONS\\_AVAILABLE](#) ([/reference/android/content/Intent.html#ACTION\\_EXTERNAL\\_APPLICATIONS\\_AVAILABLE](#)), [ACTION\\_EXTERNAL\\_APPLICATIONS\\_UNAVAILABLE](#) ([/reference/android/content/Intent.html#ACTION\\_EXTERNAL\\_APPLICATIONS\\_UNAVAILABLE](#)) and contains an integer array of uids of all of the components that have changed.

Constant Value: "android.intent.extra.changed\_uid\_list"

public static final **String** **EXTRA\_DATA\_REMOVED** Added in [API level 3](#)

Used as a boolean extra field in [ACTION\\_PACKAGE\\_REMOVED](#) ([/reference/android/content/Intent.html#ACTION\\_PACKAGE\\_REMOVED](#)) intents to indicate whether this represents a full uninstall (removing both the code and its data) or a partial uninstall (leaving its data, implying that this is an update).

Constant Value: "android.intent.extra.DATA\_REMOVED"

public static final **String** **EXTRA\_DOCK\_STATE** Added in [API level 5](#)

Used as an int extra field in [ACTION\\_DOCK\\_EVENT](#) ([/reference/android/content/Intent.html#ACTION\\_DOCK\\_EVENT](#)) intents to request the dock state. Possible values are [EXTRA\\_DOCK\\_STATE\\_UNDOCKED](#) ([/reference/android/content/Intent.html#EXTRA\\_DOCK\\_STATE\\_UNDOCKED](#)), [EXTRA\\_DOCK\\_STATE\\_DESK](#) ([/reference/android/content/Intent.html#EXTRA\\_DOCK\\_STATE\\_DESK](#)), or [EXTRA\\_DOCK\\_STATE\\_CAR](#) ([/reference/android/content/Intent.html#EXTRA\\_DOCK\\_STATE\\_CAR](#)), or [EXTRA\\_DOCK\\_STATE\\_LE\\_DESK](#) ([/reference/android/content/Intent.html#EXTRA\\_DOCK\\_STATE\\_LE\\_DESK](#)), or [EXTRA\\_DOCK\\_STATE\\_HE\\_DESK](#) ([/reference/android/content/Intent.html#EXTRA\\_DOCK\\_STATE\\_HE\\_DESK](#)).

Constant Value: "android.intent.extra.DOCK\_STATE"

public static final **int** **EXTRA\_DOCK\_STATE\_CAR** Added in [API level 5](#)

Used as an int value for [EXTRA\\_DOCK\\_STATE](#) ([/reference/android/content/Intent.html#EXTRA\\_DOCK\\_STATE](#)) to represent that the phone is in a car dock.

Constant Value: 2 (0x00000002)

public static final **int** **EXTRA\_DOCK\_STATE\_DESK** Added in [API level 5](#)

Used as an int value for [EXTRA\\_DOCK\\_STATE](#) ([/reference/android/content/Intent.html#EXTRA\\_DOCK\\_STATE](#)) to represent that the phone is in a desk dock.

Constant Value: 1 (0x00000001)

public static final int **EXTRA\_DOCK\_STATE\_HE\_DESK**

Added in [API level 11](#)

Used as an int value for [EXTRA\\_DOCK\\_STATE](#) ([/reference/android/content/Intent.html#EXTRA\\_DOCK\\_STATE](#)) to represent that the phone is in a digital (high end) dock.

Constant Value: 4 (0x00000004)

public static final int **EXTRA\_DOCK\_STATE\_LE\_DESK**

Added in [API level 11](#)

Used as an int value for [EXTRA\\_DOCK\\_STATE](#) ([/reference/android/content/Intent.html#EXTRA\\_DOCK\\_STATE](#)) to represent that the phone is in a analog (low end) dock.

Constant Value: 3 (0x00000003)

public static final int **EXTRA\_DOCK\_STATE\_UNDOCKED**

Added in [API level 5](#)

Used as an int value for [EXTRA\\_DOCK\\_STATE](#) ([/reference/android/content/Intent.html#EXTRA\\_DOCK\\_STATE](#)) to represent that the phone is not in any dock.

Constant Value: 0 (0x00000000)

public static final [String](#) **EXTRA\_DONT\_KILL\_APP**

Added in [API level 1](#)

Used as a boolean extra field in [ACTION\\_PACKAGE\\_REMOVED](#) ([/reference/android/content/Intent.html#ACTION\\_PACKAGE\\_REMOVED](#)) or [ACTION\\_PACKAGE\\_CHANGED](#) ([/reference/android/content/Intent.html#ACTION\\_PACKAGE\\_CHANGED](#)) intents to override the default action of restarting the application.

Constant Value: "android.intent.extra.DONT\_KILL\_APP"

public static final [String](#) **EXTRA\_EMAIL**

Added in [API level 1](#)

A [String\[\]](#) holding e-mail addresses that should be delivered to.

Constant Value: "android.intent.extra.EMAIL"

public static final [String](#) **EXTRA\_HTML\_TEXT**

Added in [API level 16](#)

A constant [String](#) that is associated with the Intent, used with [ACTION\\_SEND](#) ([/reference/android/content/Intent.html#ACTION\\_SEND](#)) to supply an alternative to [EXTRA\\_TEXT](#) ([/reference/android/content/Intent.html#EXTRA\\_TEXT](#)) as HTML formatted text. Note that you *must* also supply [EXTRA\\_TEXT](#) ([/reference/android/content/Intent.html#EXTRA\\_TEXT](#)).

Constant Value: "android.intent.extra.HTML\_TEXT"

public static final [String](#) **EXTRA\_INITIAL\_INTENTS**

Added in [API level 5](#)

A [Parcelable\[\]](#) of [Intent](#) ([/reference/android/content/Intent.html](#)) or [LabeledIntent](#) ([/reference/android/content/pm/LabeledIntent.html](#)) objects as set with [putExtra\(String, Parcelable\[\]\)](#) ([/reference/android/content/Intent.html#putExtra\(java.lang.String, android.os.Parcelable\[\]\)](#)) of additional activities to place a the front of the list of choices, when shown to the user with a [ACTION\\_CHOOSER](#) ([/reference/android/content/Intent.html#ACTION\\_CHOOSER](#)).

Constant Value: "android.intent.extra.INITIAL\_INTENTS"

public static final [String](#) **EXTRA\_INSTALLER\_PACKAGE\_NAME**

Added in [API level 14](#)

Used as a string extra field with [ACTION\\_INSTALL\\_PACKAGE](#) ([/reference/android/content/Intent.html#ACTION\\_INSTALL\\_PACKAGE](#)) to install a package. Specifies the installer package name; this package will receive the [ACTION\\_APP\\_ERROR](#) ([/reference/android/content/Intent.html#ACTION\\_APP\\_ERROR](#)) intent.

Constant Value: "android.intent.extra.INSTALLER\_PACKAGE\_NAME"

public static final [String](#) **EXTRA\_INTENT**

Added in [API level 1](#)

An Intent describing the choices you would like shown with [ACTION\\_PICK\\_ACTIVITY](#) ([/reference/android/content/Intent.html#ACTION\\_PICK\\_ACTIVITY](#)).

Constant Value: "android.intent.extra.INTENT"

public static final [String](#) **EXTRA\_KEY\_EVENT**

Added in [API level 1](#)

A [KeyEvent](#) ([/reference/android/view/KeyEvent.html](#)) object containing the event that triggered the creation of the Intent it is in.

Constant Value: "android.intent.extra.KEY\_EVENT"

public static final [String](#) **EXTRA\_LOCAL\_ONLY**

Added in [API level 11](#)

Extra used to indicate that an intent should only return data that is on the local device. This is a boolean extra; the default is false. If true, an implementation should only allow the user to select data that is already on the device, not requiring it be downloaded from a remote service when opened.

#### See Also

[ACTION\\_GET\\_CONTENT](#)

[ACTION\\_OPEN\\_DOCUMENT](#)

[ACTION\\_CREATE\\_DOCUMENT](#)

Constant Value: "android.intent.extra.LOCAL\_ONLY"

public static final [String](#) **EXTRA\_MIME\_TYPES**

Added in [API level 19](#)

Extra used to communicate a set of acceptable MIME types. The type of the extra is `String[]`. Values may be a combination of concrete MIME types (such as "image/png") and/or partial MIME types (such as "audio/\*").

#### See Also

[ACTION\\_GET\\_CONTENT](#)

[ACTION\\_OPEN\\_DOCUMENT](#)

Constant Value: "android.intent.extra.MIME\_TYPES"

public static final [String](#) **EXTRA\_NOT\_UNKNOWN\_SOURCE**

Added in [API level 14](#)

Used as a boolean extra field with [ACTION\\_INSTALL\\_PACKAGE](#) ([/reference/android/content/Intent.html#ACTION\\_INSTALL\\_PACKAGE](#)) to install a package. Specifies that the application being installed should not be treated as coming from an unknown source, but as coming from the app invoking the Intent. For this to work you must start the installer with `startActivityForResult()`.

Constant Value: "android.intent.extra.NOT\_UNKNOWN\_SOURCE"

public static final [String](#) **EXTRA\_ORIGINATING\_URI**

Added in [API level 17](#)

Used as a URI extra field with [ACTION\\_INSTALL\\_PACKAGE](#) ([/reference/android/content/Intent.html#ACTION\\_INSTALL\\_PACKAGE](#)) and [ACTION\\_VIEW](#) ([/reference/android/content/Intent.html#ACTION\\_VIEW](#)) to indicate the URI from which the local APK in the Intent data

field originated from.

Constant Value: "android.intent.extra.ORIGINATING\_URI"

#### public static final String **EXTRA\_PHONE\_NUMBER**

Added in [API level 1](#)

A String holding the phone number originally entered in [ACTION\\_NEW\\_OUTGOING\\_CALL](#) ([/reference/android/content/Intent.html#ACTION\\_NEW\\_OUTGOING\\_CALL](#)), or the actual number to call in a [ACTION\\_CALL](#) ([/reference/android/content/Intent.html#ACTION\\_CALL](#)).

Constant Value: "android.intent.extra.PHONE\_NUMBER"

#### public static final String **EXTRA\_REFERRER**

Added in [API level 17](#)

Used as a URI extra field with [ACTION\\_INSTALL\\_PACKAGE](#) ([/reference/android/content/Intent.html#ACTION\\_INSTALL\\_PACKAGE](#)) and [ACTION\\_VIEW](#) ([/reference/android/content/Intent.html#ACTION\\_VIEW](#)) to indicate the HTTP referrer URI associated with the Intent data field or [EXTRA\\_ORIGINATING\\_URI](#) ([/reference/android/content/Intent.html#EXTRA\\_ORIGINATING\\_URI](#)).

Constant Value: "android.intent.extra.REFERRER"

#### public static final String **EXTRA\_REMOTE\_INTENT\_TOKEN**

Added in [API level 5](#)

Used in the extra field in the remote intent. It's a string token passed with the remote intent.

Constant Value: "android.intent.extra.remote\_intent\_token"

#### public static final String **EXTRA\_REPLACING**

Added in [API level 3](#)

Used as a boolean extra field in [ACTION\\_PACKAGE\\_REMOVED](#) ([/reference/android/content/Intent.html#ACTION\\_PACKAGE\\_REMOVED](#)) intents to indicate that this is a replacement of the package, so this broadcast will immediately be followed by an add broadcast for a different version of the same package.

Constant Value: "android.intent.extra.REPLACING"

#### public static final String **EXTRA\_RESTRICTIONS\_BUNDLE**

Added in [API level 18](#)

Extra sent in the intent to the BroadcastReceiver that handles [ACTION\\_GET\\_RESTRICTION\\_ENTRIES](#) ([/reference/android/content/Intent.html#ACTION\\_GET\\_RESTRICTION\\_ENTRIES](#)). The type of the extra is a Bundle containing the restrictions as key/value pairs.

Constant Value: "android.intent.extra.restrictions\_bundle"

#### public static final String **EXTRA\_RESTRICTIONS\_INTENT**

Added in [API level 18](#)

Extra used in the response from a BroadcastReceiver that handles [ACTION\\_GET\\_RESTRICTION\\_ENTRIES](#) ([/reference/android/content/Intent.html#ACTION\\_GET\\_RESTRICTION\\_ENTRIES](#)).

Constant Value: "android.intent.extra.restrictions\_intent"

#### public static final String **EXTRA\_RESTRICTIONS\_LIST**

Added in [API level 18](#)

Extra used in the response from a BroadcastReceiver that handles [ACTION\\_GET\\_RESTRICTION\\_ENTRIES](#) ([/reference/android/content/Intent.html#ACTION\\_GET\\_RESTRICTION\\_ENTRIES](#)). The type of the extra is `ArrayList<RestrictionEntry>`.



Constant Value: "android.intent.extra.restrictions\_list"

public static final String **EXTRA\_RETURN\_RESULT**

Added in API level 14

Used as a boolean extra field with [ACTION\\_INSTALL\\_PACKAGE](#) ([/reference/android/content/Intent.html#ACTION\\_INSTALL\\_PACKAGE](#)) or [ACTION\\_UNINSTALL\\_PACKAGE](#) ([/reference/android/content/Intent.html#ACTION\\_UNINSTALL\\_PACKAGE](#)). Specifies that the installer UI should return to the application the result code of the install/uninstall. The returned result code will be [RESULT\\_OK](#) ([/reference/android/app/Activity.html#RESULT\\_OK](#)) on success or [RESULT\\_FIRST\\_USER](#) ([/reference/android/app/Activity.html#RESULT\\_FIRST\\_USER](#)) on failure.

Constant Value: "android.intent.extra.RETURN\_RESULT"

public static final String **EXTRA\_SHORTCUT\_ICON**

Added in API level 1

The name of the extra used to define the icon, as a Bitmap, of a shortcut.

**See Also**

[ACTION\\_CREATE\\_SHORTCUT](#)

Constant Value: "android.intent.extra.shortcut.ICON"

public static final String **EXTRA\_SHORTCUT\_ICON\_RESOURCE**

Added in API level 1

The name of the extra used to define the icon, as a ShortcutIconResource, of a shortcut.

**See Also**

[ACTION\\_CREATE\\_SHORTCUT](#)

[Intent.ShortcutIconResource](#)

Constant Value: "android.intent.extra.shortcut.ICON\_RESOURCE"

public static final String **EXTRA\_SHORTCUT\_INTENT**

Added in API level 1

The name of the extra used to define the Intent of a shortcut.

**See Also**

[ACTION\\_CREATE\\_SHORTCUT](#)

Constant Value: "android.intent.extra.shortcut.INTENT"

public static final String **EXTRA\_SHORTCUT\_NAME**

Added in API level 1

The name of the extra used to define the name of a shortcut.

**See Also**

[ACTION\\_CREATE\\_SHORTCUT](#)

Constant Value: "android.intent.extra.shortcut.NAME"

public static final String **EXTRA\_SHUTDOWN\_USERSPACE\_ONLY**

Added in API level 19

Optional extra for [ACTION\\_SHUTDOWN](#) ([/reference/android/content/Intent.html#ACTION\\_SHUTDOWN](#)) that allows the sender to qualify that this shutdown is only for the user space of the system, not a complete shutdown. When this is true, hardware devices can use this information to determine that they shouldn't do a complete shutdown of their device since this is not a complete shutdown down to the kernel, but only user space restarting. The default if not supplied is false.

Constant Value: "android.intent.extra.SHUTDOWN\_USERSPACE\_ONLY"

public static final String **EXTRA\_STREAM**

Added in API level 1

A content: URI holding a stream of data associated with the Intent, used with [ACTION\\_SEND](/reference/android/content/Intent.html#ACTION_SEND) ([/reference/android/content/Intent.html#ACTION\\_SEND](/reference/android/content/Intent.html#ACTION_SEND)) to supply the data being sent.

Constant Value: "android.intent.extra.STREAM"

public static final **String** **EXTRA\_SUBJECT**

Added in [API level 1](#)

A constant string holding the desired subject line of a message.

Constant Value: "android.intent.extra.SUBJECT"

public static final **String** **EXTRA\_TEMPLATE**

Added in [API level 1](#)

The initial data to place in a newly created record. Use with [ACTION\\_INSERT](/reference/android/content/Intent.html#ACTION_INSERT) ([/reference/android/content/Intent.html#ACTION\\_INSERT](/reference/android/content/Intent.html#ACTION_INSERT)). The data here is a Map containing the same fields as would be given to the underlying `ContentProvider.insert()` call.

Constant Value: "android.intent.extra.TEMPLATE"

public static final **String** **EXTRA\_TEXT**

Added in [API level 1](#)

A constant CharSequence that is associated with the Intent, used with [ACTION\\_SEND](/reference/android/content/Intent.html#ACTION_SEND) ([/reference/android/content/Intent.html#ACTION\\_SEND](/reference/android/content/Intent.html#ACTION_SEND)) to supply the literal data to be sent.

Note that this may be a styled CharSequence, so you must use

[Bundle.getCharSequence\(\)](/reference/android/os/Bundle.html#getCharSequence(java.lang.String)) ([/reference/android/os/Bundle.html#getCharSequence\(java.lang.String\)](/reference/android/os/Bundle.html#getCharSequence(java.lang.String))) to retrieve it.

Constant Value: "android.intent.extra.TEXT"

public static final **String** **EXTRA\_TITLE**

Added in [API level 1](#)

A CharSequence dialog title to provide to the user when used with a [ACTION\\_CHOOSER](/reference/android/content/Intent.html#ACTION_CHOOSER) ([/reference/android/content/Intent.html#ACTION\\_CHOOSER](/reference/android/content/Intent.html#ACTION_CHOOSER)).

Constant Value: "android.intent.extra.TITLE"

public static final **String** **EXTRA\_UID**

Added in [API level 1](#)

Used as an int extra field in [ACTION\\_UID\\_REMOVED](/reference/android/content/Intent.html#ACTION_UID_REMOVED) ([/reference/android/content/Intent.html#ACTION\\_UID\\_REMOVED](/reference/android/content/Intent.html#ACTION_UID_REMOVED)) intents to supply the uid the package had been assigned.

Also an optional extra in [ACTION\\_PACKAGE\\_REMOVED](/reference/android/content/Intent.html#ACTION_PACKAGE_REMOVED) ([/reference/android/content/Intent.html#ACTION\\_PACKAGE\\_REMOVED](/reference/android/content/Intent.html#ACTION_PACKAGE_REMOVED)) or [ACTION\\_PACKAGE\\_CHANGED](/reference/android/content/Intent.html#ACTION_PACKAGE_CHANGED) ([/reference/android/content/Intent.html#ACTION\\_PACKAGE\\_CHANGED](/reference/android/content/Intent.html#ACTION_PACKAGE_CHANGED)) for the same purpose.

Constant Value: "android.intent.extra.UID"

public static final **int** **FILL\_IN\_ACTION**

Added in [API level 1](#)

Use with [fillIn\(Intent, int\)](/reference/android/content/Intent.html#fillIn(android.content.Intent, int)) ([/reference/android/content/Intent.html#fillIn\(android.content.Intent, int\)](/reference/android/content/Intent.html#fillIn(android.content.Intent, int))) to allow the current action value to be overwritten, even if it is already set.

Constant Value: 1 (0x00000001)

public static final **int** **FILL\_IN\_CATEGORIES**

Added in [API level 1](#)

Use with [fillIn\(Intent, int\)](/reference/android/content/Intent.html#fillIn(android.content.Intent, int)) ([/reference/android/content/Intent.html#fillIn\(android.content.Intent, int\)](/reference/android/content/Intent.html#fillIn(android.content.Intent, int))) to allow the current categories to be overwritten, even if they are already set.

Constant Value: 4 (0x00000004)

### public static final int **FILL\_IN\_CLIP\_DATA**

Added in [API level 16](#)

Use with [fillIn\(Intent, int\)](#) ([/reference/android/content/Intent.html#fillIn\(android.content.Intent, int\)](#)) to allow the current ClipData to be overwritten, even if it is already set.

Constant Value: 128 (0x00000080)

### public static final int **FILL\_IN\_COMPONENT**

Added in [API level 1](#)

Use with [fillIn\(Intent, int\)](#) ([/reference/android/content/Intent.html#fillIn\(android.content.Intent, int\)](#)) to allow the current component value to be overwritten, even if it is already set.

Constant Value: 8 (0x00000008)

### public static final int **FILL\_IN\_DATA**

Added in [API level 1](#)

Use with [fillIn\(Intent, int\)](#) ([/reference/android/content/Intent.html#fillIn\(android.content.Intent, int\)](#)) to allow the current data or type value overwritten, even if it is already set.

Constant Value: 2 (0x00000002)

### public static final int **FILL\_IN\_PACKAGE**

Added in [API level 4](#)

Use with [fillIn\(Intent, int\)](#) ([/reference/android/content/Intent.html#fillIn\(android.content.Intent, int\)](#)) to allow the current package value to be overwritten, even if it is already set.

Constant Value: 16 (0x00000010)

### public static final int **FILL\_IN\_SELECTOR**

Added in [API level 15](#)

Use with [fillIn\(Intent, int\)](#) ([/reference/android/content/Intent.html#fillIn\(android.content.Intent, int\)](#)) to allow the current selector to be overwritten, even if it is already set.

Constant Value: 64 (0x00000040)

### public static final int **FILL\_IN\_SOURCE\_BOUNDS**

Added in [API level 7](#)

Use with [fillIn\(Intent, int\)](#) ([/reference/android/content/Intent.html#fillIn\(android.content.Intent, int\)](#)) to allow the current bounds rectangle to be overwritten, even if it is already set.

Constant Value: 32 (0x00000020)

### public static final int **FLAG\_ACTIVITY\_BROUGHT\_TO\_FRONT**

Added in [API level 1](#)

This flag is not normally set by application code, but set for you by the system as described in the [launchMode](#) ([/reference/android/R.styleable.html#AndroidManifestActivity launchMode](#)) documentation for the singleTask mode.

Constant Value: 4194304 (0x00400000)

### public static final int **FLAG\_ACTIVITY\_CLEAR\_TASK**

Added in [API level 11](#)

If set in an Intent passed to `Context.startActivity()` ([/reference/android/content/Context.html#startActivity\(android.content.Intent\)](/reference/android/content/Context.html#startActivity(android.content.Intent))), 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 only be used in conjunction with `FLAG_ACTIVITY_NEW_TASK` ([/reference/android/content/Intent.html#FLAG\\_ACTIVITY\\_NEW\\_TASK](/reference/android/content/Intent.html#FLAG_ACTIVITY_NEW_TASK)).

Constant Value: 32768 (0x00008000)

#### public static final int **FLAG\_ACTIVITY\_CLEAR\_TOP**

Added in [API level 1](#)

If set, and the activity being launched is already running in the current task, then instead of launching a new instance of that activity, all of the other activities on top of it will be closed and this Intent will be delivered to the (now on top) old activity as a new Intent.

For example, consider a task consisting of the activities: A, B, C, D. If D calls `startActivity()` with an Intent that resolves to the component of activity B, then C and D will be finished and B receive the given Intent, resulting in the stack now being: A, B.

The currently running instance of activity B in the above example will either receive the new intent you are starting here in its `onNewIntent()` method, or be itself finished and restarted with the new intent. If it has declared its launch mode to be "multiple" (the default) and you have not set `FLAG_ACTIVITY_SINGLE_TOP` ([/reference/android/content/Intent.html#FLAG\\_ACTIVITY\\_SINGLE\\_TOP](/reference/android/content/Intent.html#FLAG_ACTIVITY_SINGLE_TOP)) in the same intent, then it will be finished and re-created; for all other launch modes or if `FLAG_ACTIVITY_SINGLE_TOP` ([/reference/android/content/Intent.html#FLAG\\_ACTIVITY\\_SINGLE\\_TOP](/reference/android/content/Intent.html#FLAG_ACTIVITY_SINGLE_TOP)) is set then this Intent will be delivered to the current instance's `onNewIntent()`.

This launch mode can also be used to good effect in conjunction with `FLAG_ACTIVITY_NEW_TASK` ([/reference/android/content/Intent.html#FLAG\\_ACTIVITY\\_NEW\\_TASK](/reference/android/content/Intent.html#FLAG_ACTIVITY_NEW_TASK)): if used to start the root activity of a task, it will bring any currently running instance of that task to the foreground, and then clear it to its root state. This is especially useful, for example, when launching an activity from the notification manager.

See [Tasks and Back Stack](/guide/topics/fundamentals/tasks-and-back-stack.html) for more information about tasks.

Constant Value: 67108864 (0x04000000)

#### public static final int **FLAG\_ACTIVITY\_CLEAR\_WHEN\_TASK\_RESET**

Added in [API level 3](#)

If set, this marks a point in the task's activity stack that should be cleared when the task is reset. That is, the next time the task is brought to the foreground with `FLAG_ACTIVITY_RESET_TASK_IF_NEEDED` ([/reference/android/content/Intent.html#FLAG\\_ACTIVITY\\_RESET\\_TASK\\_IF\\_NEEDED](/reference/android/content/Intent.html#FLAG_ACTIVITY_RESET_TASK_IF_NEEDED)) (typically as a result of the user re-launching it from home), this activity and all on top of it will be finished so that the user does not return to them, but instead returns to whatever activity preceeded it.

This is useful for cases where you have a logical break in your application. For example, an e-mail application may have a command to view an attachment, which launches an image view activity to display it. This activity should be part of the e-mail application's task, since it is a part of the task the user is involved in. However, if the user leaves that task, and later selects the e-mail app from home, we may like them to return to the conversation they were viewing, not the picture attachment, since that is confusing. By setting this flag when launching the image viewer, that viewer and any activities it starts will be removed the next time the user returns to mail.

Constant Value: 524288 (0x00080000)

#### public static final int **FLAG\_ACTIVITY\_EXCLUDE\_FROM\_RECENTS**

Added in [API level 1](#)

If set, the new activity is not kept in the list of recently launched activities.

Constant Value: 8388608 (0x00800000)

#### public static final int **FLAG\_ACTIVITY\_FORWARD\_RESULT**

Added in [API level 1](#)

If set and this intent is being used to launch a new activity from an existing one, then the reply target of the existing activity will be transferred to the new activity. This way the new activity can call `setResult(int)` ([/reference/android/app/Activity.html#setResult\(int\)](/reference/android/app/Activity.html#setResult(int))) and have that result sent back to the reply target of the original activity.

Constant Value: 33554432 (0x02000000)

#### public static final int **FLAG\_ACTIVITY\_LAUNCHED\_FROM\_HISTORY**

Added in [API level 1](#)

This flag is not normally set by application code, but set for you by the system if this activity is being launched from history (longpress home key).

Constant Value: 1048576 (0x00100000)

#### public static final int **FLAG\_ACTIVITY\_MULTIPLE\_TASK**

Added in [API level 1](#)

**Do not use this flag unless you are implementing your own top-level application launcher.** Used in conjunction with `FLAG_ACTIVITY_NEW_TASK` ([/reference/android/content/Intent.html#FLAG\\_ACTIVITY\\_NEW\\_TASK](/reference/android/content/Intent.html#FLAG_ACTIVITY_NEW_TASK)) to disable the behavior of bringing an existing task to the foreground. When set, a new task is *always* started to host the Activity for the Intent, regardless of whether there is already an existing task running the same thing.

**Because the default system does not include graphical task management, you should not use this flag unless you provide some way for a user to return back to the tasks you have launched.**

This flag is ignored if `FLAG_ACTIVITY_NEW_TASK` ([/reference/android/content/Intent.html#FLAG\\_ACTIVITY\\_NEW\\_TASK](/reference/android/content/Intent.html#FLAG_ACTIVITY_NEW_TASK)) is not set.

See [Tasks and Back Stack](/guide/topics/fundamentals/tasks-and-back-stack.html) (</guide/topics/fundamentals/tasks-and-back-stack.html>) for more information about tasks.

Constant Value: 134217728 (0x08000000)

#### public static final int **FLAG\_ACTIVITY\_NEW\_TASK**

Added in [API level 1](#)

If set, this activity will become the start of a new task on this history stack. A task (from the activity that started it to the next task activity) defines an atomic group of activities that the user can move to. Tasks can be moved to the foreground and background; all of the activities inside of a particular task always remain in the same order. See [Tasks and Back Stack](/guide/topics/fundamentals/tasks-and-back-stack.html) (</guide/topics/fundamentals/tasks-and-back-stack.html>) for more information about tasks.

This flag is generally used by activities that want to present a "launcher" style behavior: they give the user a list of separate things that can be done, which otherwise run completely independently of the activity launching them.

When using this flag, if a task is already running for the activity you are now starting, then a new activity will not be started; instead, the current task will simply be brought to the front of the screen with the state it was last in. See [FLAG\\_ACTIVITY\\_MULTIPLE\\_TASK](/reference/android/content/Intent.html#FLAG_ACTIVITY_MULTIPLE_TASK) ([/reference/android/content/Intent.html#FLAG\\_ACTIVITY\\_MULTIPLE\\_TASK](/reference/android/content/Intent.html#FLAG_ACTIVITY_MULTIPLE_TASK)) for a flag to disable this behavior.

This flag can not be used when the caller is requesting a result from the activity being launched.

Constant Value: 268435456 (0x10000000)

**public static final int FLAG\_ACTIVITY\_NO\_ANIMATION**Added in [API level 5](#)

If set in an Intent passed to `Context.startActivity()` ([/reference/android/content/Context.html#startActivity\(android.content.Intent\)](#)), this flag will prevent the system from applying an activity transition animation to go to the next activity state. This doesn't mean an animation will never run – if another activity change happens that doesn't specify this flag before the activity started here is displayed, then that transition will be used. This flag can be put to good use when you are going to do a series of activity operations but the animation seen by the user shouldn't be driven by the first activity change but rather a later one.

Constant Value: 65536 (0x00010000)

**public static final int FLAG\_ACTIVITY\_NO\_HISTORY**Added in [API level 1](#)

If set, the new activity is not kept in the history stack. As soon as the user navigates away from it, the activity is finished. This may also be set with the `noHistory` ([/reference/android/R.styleable.html#AndroidManifestActivity\\_noHistory](#)) attribute.

Constant Value: 1073741824 (0x40000000)

**public static final int FLAG\_ACTIVITY\_NO\_USER\_ACTION**Added in [API level 3](#)

If set, this flag will prevent the normal `onUserLeaveHint()` ([/reference/android/app/Activity.html#onUserLeaveHint\(\)](#)) callback from occurring on the current frontmost activity before it is paused as the newly-started activity is brought to the front.

Typically, an activity can rely on that callback to indicate that an explicit user action has caused their activity to be moved out of the foreground. The callback marks an appropriate point in the activity's lifecycle for it to dismiss any notifications that it intends to display "until the user has seen them," such as a blinking LED.

If an activity is ever started via any non-user-driven events such as phone-call receipt or an alarm handler, this flag should be passed to `Context.startActivity` ([/reference/android/content/Context.html#startActivity\(android.content.Intent\)](#)), ensuring that the pausing activity does not think the user has acknowledged its notification.

Constant Value: 262144 (0x00040000)

**public static final int FLAG\_ACTIVITY\_PREVIOUS\_IS\_TOP**Added in [API level 1](#)

If set and this intent is being used to launch a new activity from an existing one, 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 one. The previous activity will be used as the top, with the assumption being that the current activity will finish itself immediately.

Constant Value: 16777216 (0x01000000)

**public static final int FLAG\_ACTIVITY\_REORDER\_TO\_FRONT**Added in [API level 3](#)

If set in an Intent passed to `Context.startActivity()` ([/reference/android/content/Context.html#startActivity\(android.content.Intent\)](#)), this flag will cause the launched activity to be brought to the front of its task's history stack if it is already running.

For example, consider a task consisting of four activities: A, B, C, D. If D calls `startActivity()` with an Intent that resolves to the component of activity B, then B will be brought to the front of the history stack, with this resulting order: A, C, D, B. This flag will be ignored if `FLAG_ACTIVITY_CLEAR_TOP` ([/reference/android/content/Intent.html#FLAG\\_ACTIVITY\\_CLEAR\\_TOP](#)) is also specified.

Constant Value: 131072 (0x00020000)

**public static final int FLAG\_ACTIVITY\_RESET\_TASK\_IF\_NEEDED** Added in [API level 1](#)

If set, and this activity is either being started in a new task or bringing to the top an existing task, then it will be launched as the front door of the task. This will result in the application of any affinities needed to have that task in the proper state (either moving activities to or from it), or simply resetting that task to its initial state if needed.

Constant Value: 2097152 (0x00200000)

**public static final int FLAG\_ACTIVITY\_SINGLE\_TOP** Added in [API level 1](#)

If set, the activity will not be launched if it is already running at the top of the history stack.

Constant Value: 536870912 (0x20000000)

**public static final int FLAG\_ACTIVITY\_TASK\_ON\_HOME** Added in [API level 11](#)

If set in an Intent passed to `Context.startActivity()` ([/reference/android/content/Context.html#startActivity\(android.content.Intent\)](#)), this flag will cause a newly launching task to be placed on top of the current home activity task (if there is one). That is, pressing back from the task will always return the user to home even if that was not the last activity they saw. This can only be used in conjunction with [FLAG\\_ACTIVITY\\_NEW\\_TASK](#) ([/reference/android/content/Intent.html#FLAG\\_ACTIVITY\\_NEW\\_TASK](#)).

Constant Value: 16384 (0x00004000)

**public static final int FLAG\_DEBUG\_LOG\_RESOLUTION** Added in [API level 1](#)

A flag you can enable 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.

Constant Value: 8 (0x00000008)

**public static final int FLAG\_EXCLUDE\_STOPPED\_PACKAGES** Added in [API level 12](#)

If set, this intent will not match any components in packages that are currently stopped. If this is not set, then the default behavior is to include such applications in the result.

Constant Value: 16 (0x00000010)

**public static final int FLAG\_FROM\_BACKGROUND** Added in [API level 1](#)

Can be set by the caller to indicate that this Intent is coming from a background operation, not from direct user interaction.

Constant Value: 4 (0x00000004)

**public static final int FLAG\_GRANT\_PERSISTABLE\_URI\_PERMISSION** Added in [API level 19](#)

When combined with [FLAG\\_GRANT\\_READ\\_URI\\_PERMISSION](#) ([/reference/android/content/Intent.html#FLAG\\_GRANT\\_READ\\_URI\\_PERMISSION](#)) and/or [FLAG\\_GRANT\\_WRITE\\_URI\\_PERMISSION](#) ([/reference/android/content/Intent.html#FLAG\\_GRANT\\_WRITE\\_URI\\_PERMISSION](#)), the URI permission grant can be persisted across device reboots until explicitly revoked with [revokeUriPermission\(Uri, int\)](#) ([/reference/android/content/Context.html#revokeUriPermission\(android.net.Uri, int\)](#)). This flag only offers the grant for possible persisting; the receiving application must call [takePersistableUriPermission\(Uri, int\)](#) ([/reference/android/content/ContentResolver.html#takePersistableUriPermission\(android.net.Uri, int\)](#)) to actually persist.

#### See Also

[takePersistableUriPermission\(Uri, int\)](#)

```
releasePersistableUriPermission(Uri, int)  
getPersistedUriPermissions()  
getOutgoingPersistedUriPermissions()
```

Constant Value: 64 (0x00000040)

#### public static final int **FLAG\_GRANT\_READ\_URI\_PERMISSION**

Added in [API level 1](#)

If set, the recipient of this Intent will be granted permission to perform read operations on the URI in the Intent's data and any URIs specified in its ClipData. When applying to an Intent's ClipData, all URIs as well as recursive traversals through data or other ClipData in Intent items will be granted; only the grant flags of the top-level Intent are used.

Constant Value: 1 (0x00000001)

#### public static final int **FLAG\_GRANT\_WRITE\_URI\_PERMISSION**

Added in [API level 1](#)

If set, the recipient of this Intent will be granted permission to perform write operations on the URI in the Intent's data and any URIs specified in its ClipData. When applying to an Intent's ClipData, all URIs as well as recursive traversals through data or other ClipData in Intent items will be granted; only the grant flags of the top-level Intent are used.

Constant Value: 2 (0x00000002)

#### public static final int **FLAG\_INCLUDE\_STOPPED\_PACKAGES**

Added in [API level 12](#)

If set, this intent will always match any components in packages that are currently stopped. This is the default behavior when [`FLAG\_EXCLUDE\_STOPPED\_PACKAGES`](#) ([/reference/android/content/Intent.html#FLAG\\_EXCLUDE\\_STOPPED\\_PACKAGES](#)) is not set. If both of these flags are set, this one wins (it allows overriding of exclude for places where the framework may automatically set the exclude flag).

Constant Value: 32 (0x00000020)

#### public static final int **FLAG\_RECEIVER\_FOREGROUND**

Added in [API level 16](#)

If set, when sending a broadcast the recipient is allowed to run at foreground priority, with a shorter timeout interval. During normal broadcasts the receivers are not automatically hoisted out of the background priority class.

Constant Value: 268435456 (0x10000000)

#### public static final int **FLAG\_RECEIVER\_NO\_ABORT**

Added in [API level 19](#)

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 can not prevent later receivers from seeing the broadcast.

Constant Value: 134217728 (0x08000000)

#### public static final int **FLAG\_RECEIVER\_REGISTERED\_ONLY**

Added in [API level 1](#)

If set, when sending a broadcast only registered receivers will be called -- no BroadcastReceiver components will be launched.

Constant Value: 1073741824 (0x40000000)

#### public static final int **FLAG\_RECEIVER\_REPLACE\_PENDING**

Added in [API level 8](#)

If set, when sending a broadcast the new broadcast will replace any existing pending broadcast that matches it. Matching is defined by [Intent.filterEquals](#) ([/reference/android/content/Intent.html#filterEquals\(android.content.Intent\)](#)) returning true for the intents of the two broadcasts. When a match is found, the new broadcast (and receivers



associated with it) will replace the existing one in the pending broadcast list, remaining at the same position in the list.

This flag is most typically used with sticky broadcasts, which only care about delivering the most recent values of the broadcast to their receivers.

Constant Value: 536870912 (0x20000000)

public static final String **METADATA\_DOCK\_HOME**

Added in API level 5

Boolean that can be supplied as meta-data with a dock activity, to indicate that the dock should take over the home key when it is active.

Constant Value: "android.dock\_home"

public static final int **URI\_INTENT\_SCHEME**

Added in API level 4

Flag for use with [toUri\(int\)](#) ([/reference/android/content/Intent.html#toUri\(int\)](#)) and [parseUri\(String, int\)](#) ([/reference/android/content/Intent.html#parseUri\(java.lang.String, int\)](#)): the URI string always has the "intent:" scheme. This syntax can be used when you want to later disambiguate between URIs that are intended to describe an Intent vs. all others that should be treated as raw URIs. When used with [parseUri\(String, int\)](#) ([/reference/android/content/Intent.html#parseUri\(java.lang.String, int\)](#)), any other scheme will result in a generic VIEW action for that raw URI.

Constant Value: 1 (0x00000001)

## Fields

---

public static final Creator<Intent> **CREATOR**

Added in API level 1

## Public Constructors

---

public **Intent** ()

Added in API level 1

Create an empty intent.

public **Intent** (Intent o)

Added in API level 1

Copy constructor.

public **Intent** (String action)

Added in API level 1

Create an intent with a given action. All other fields (data, type, class) are null. Note that the action *must* be in a namespace because Intents are used globally in the system -- for example the system VIEW action is android.intent.action.VIEW; an application's custom action would be something like com.google.app.myapp.CUSTOM\_ACTION.

### Parameters

*action* The Intent action, such as ACTION\_VIEW.

public **Intent** (String action, Uri uri)

Added in API level 1

Create an intent with a given action and for a given data url. Note that the action *must* be in a namespace because Intents are used globally in the system -- for example the system VIEW action is android.intent.action.VIEW; an application's custom action would be something like com.google.app.myapp.CUSTOM\_ACTION.

*Note: scheme and host name matching in the Android framework is case-sensitive, unlike the formal RFC. As a result, you should always ensure that you write your Uri with these elements using lower case letters, and normalize any Uris you receive from outside of Android to ensure the scheme and host is lower case.*

#### Parameters

*action*    The Intent action, such as ACTION\_VIEW.  
*uri*        The Intent data URI.

**public Intent** (Context packageContext, Class<?> cls)

Added in [API level 1](#)

Create an intent for a specific component. All other fields (action, data, type, class) are null, though they can be modified later with explicit calls. This provides a convenient way to create an intent that is intended to execute a hard-coded class name, rather than relying on the system to find an appropriate class for you; see [setComponent\(ComponentName\)](#) ([/reference/android/content/Intent.html#setComponent\(android.content.ComponentName\)](#)) for more information on the repercussions of this.

#### Parameters

*packageContext*    A Context of the application package implementing this class.  
*cls*                The component class that is to be used for the intent.

#### See Also

[setClass\(Context, Class\)](#)  
[setComponent\(ComponentName\)](#)  
[Intent\(String, android.net.Uri, Context, Class\)](#)

**public Intent** (String action, Uri uri, Context packageContext, Class<?> cls)

Added in [API level 1](#)

Create an intent for a specific component with a specified action and data. This is equivalent using [Intent\(String, android.net.Uri\)](#) ([/reference/android/content/Intent.html#Intent\(java.lang.String, android.net.Uri\)](#)) to construct the Intent and then calling [setClass\(Context, Class\)](#) ([/reference/android/content/Intent.html#setClass\(android.content.Context, java.lang.Class<?>\)](#)) to set its class.

*Note: scheme and host name matching in the Android framework is case-sensitive, unlike the formal RFC. As a result, you should always ensure that you write your Uri with these elements using lower case letters, and normalize any Uris you receive from outside of Android to ensure the scheme and host is lower case.*

#### Parameters

*action*        The Intent action, such as ACTION\_VIEW.  
*uri*            The Intent data URI.  
*packageContext*    A Context of the application package implementing this class.  
*cls*            The component class that is to be used for the intent.

#### See Also

[Intent\(String, android.net.Uri\)](#)  
[Intent\(Context, Class\)](#)  
[setClass\(Context, Class\)](#)  
[setComponent\(ComponentName\)](#)

## Public Methods

---

**public Intent** [addCategory](#) (String category)

Add a new category to the intent. Categories provide additional detail about the action the intent performs. When resolving an intent, only activities that provide *all* of the requested categories will be used.

#### Parameters

*category* The desired category. This can be either one of the predefined Intent categories, or a custom category in your own namespace.

#### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

#### See Also

[hasCategory\(String\)](#)  
[removeCategory\(String\)](#)

public [Intent](#) **addFlags** (int flags)

Added in [API level 1](#)

Add additional flags to the intent (or with existing flags value).

#### Parameters

*flags* The new flags to set.

#### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

#### See Also

[setFlags\(int\)](#)

public [Object](#) **clone** ()

Added in [API level 1](#)

Creates and returns a copy of this `Object`. The default implementation returns a so-called "shallow" copy: It creates a new instance of the same class and then copies the field values (including object references) from this instance to the new instance. A "deep" copy, in contrast, would also recursively clone nested objects. A subclass that needs to implement this kind of cloning should call `super.clone()` to create the new instance and then create deep copies of the nested, mutable objects.

#### Returns

a copy of this object.

public [Intent](#) **cloneFilter** ()

Added in [API level 1](#)

Make a clone of only the parts of the Intent that are relevant for filter matching: the action, data, type, component, and categories.

public static [Intent](#) **createChooser** ([Intent](#) target, [CharSequence](#) title)

Added in [API level 1](#)

Convenience function for creating a [ACTION\\_CHOOSER](#) ([/reference/android/content/Intent.html#ACTION\\_CHOOSER](#)) Intent.

Builds a new [ACTION\\_CHOOSER](#) ([/reference/android/content/Intent.html#ACTION\\_CHOOSER](#)) Intent that wraps the given target intent, also optionally supplying a title. If the target intent has specified [FLAG\\_GRANT\\_READ\\_URI\\_PERMISSION](#) ([/reference/android/content/Intent.html#FLAG\\_GRANT\\_READ\\_URI\\_PERMISSION](#)) or [FLAG\\_GRANT\\_WRITE\\_URI\\_PERMISSION](#) ([/reference/android/content/Intent.html#FLAG\\_GRANT\\_WRITE\\_URI\\_PERMISSION](#)), then these flags will also be set in the returned chooser intent, with its `ClipData` set appropriately: either a direct reflection of [getClipData\(\)](#) ([/reference/android/content/Intent.html#getClipData\(\)](#)) if that is non-null, or a new `ClipData` built from [getData\(\)](#) ([/reference/android/content/Intent.html#getData\(\)](#)).

#### Parameters

*target* The Intent that the user will be selecting an activity to perform.

*title* Optional title that will be displayed in the chooser.

**Returns**

Return a new Intent object that you can hand to [Context.startActivity\(\)](#) and related methods.

public int **describeContents** ()

Added in [API level 1](#)

Describe the kinds of special objects contained in this Parcelable's marshalled representation.

**Returns**

a bitmask indicating the set of special object types marshalled by the Parcelable.

public int **fillIn** ([Intent](#) other, int flags)

Added in [API level 1](#)

Copy the contents of *other* in to this object, but only where fields are not defined by this object. For purposes of a field being defined, the following pieces of data in the Intent are considered to be separate fields:

- action, as set by [setAction\(String\)](#).
- data Uri and MIME type, as set by [setData\(Uri\)](#), [setType\(String\)](#), or [setDataAndType\(Uri, String\)](#).
- categories, as set by [addCategory\(String\)](#).
- package, as set by [setPackage\(String\)](#).
- component, as set by [setComponent\(ComponentName\)](#) or related methods.
- source bounds, as set by [setSourceBounds\(Rect\)](#).
- selector, as set by [setSelector\(Intent\)](#).
- clip data, as set by [setClipData\(ClipData\)](#).
- each top-level name in the associated extras.

In addition, you can use the [FILL\\_IN\\_ACTION](#) ([/reference/android/content/Intent.html#FILL\\_IN\\_ACTION](#)), [FILL\\_IN\\_DATA](#) ([/reference/android/content/Intent.html#FILL\\_IN\\_DATA](#)), [FILL\\_IN\\_CATEGORIES](#) ([/reference/android/content/Intent.html#FILL\\_IN\\_CATEGORIES](#)), [FILL\\_IN\\_PACKAGE](#) ([/reference/android/content/Intent.html#FILL\\_IN\\_PACKAGE](#)), [FILL\\_IN\\_COMPONENT](#) ([/reference/android/content/Intent.html#FILL\\_IN\\_COMPONENT](#)), [FILL\\_IN\\_SOURCE\\_BOUNDS](#) ([/reference/android/content/Intent.html#FILL\\_IN\\_SOURCE\\_BOUNDS](#)), [FILL\\_IN\\_SELECTOR](#) ([/reference/android/content/Intent.html#FILL\\_IN\\_SELECTOR](#)), and [FILL\\_IN\\_CLIP\\_DATA](#) ([/reference/android/content/Intent.html#FILL\\_IN\\_CLIP\\_DATA](#)) to override the restriction where the corresponding field will not be replaced if it is already set.

Note: The component field will only be copied if [FILL\\_IN\\_COMPONENT](#) ([/reference/android/content/Intent.html#FILL\\_IN\\_COMPONENT](#)) is explicitly specified. The selector will only be copied if [FILL\\_IN\\_SELECTOR](#) ([/reference/android/content/Intent.html#FILL\\_IN\\_SELECTOR](#)) is explicitly specified.

For example, consider Intent A with {data="foo", categories="bar"} and Intent B with {action="gotit", data-type="some/thing", categories="one","two"}.

Calling A.fillIn(B, Intent.FILL\_IN\_DATA) will result in A now containing: {action="gotit", data-type="some/thing", categories="bar"}.

**Parameters**

*other* Another Intent whose values are to be used to fill in the current one.

*flags* Options to control which fields can be filled in.

**Returns**

Returns a bit mask of [FILL\\_IN\\_ACTION](#), [FILL\\_IN\\_DATA](#), [FILL\\_IN\\_CATEGORIES](#), [FILL\\_IN\\_PACKAGE](#), [FILL\\_IN\\_COMPONENT](#), [FILL\\_IN\\_SOURCE\\_BOUNDS](#), and [FILL\\_IN\\_SELECTOR](#) indicating which fields were changed.

public boolean **filterEquals** ([Intent](#) other)

Added in [API level 1](#)

Determine if two intents are the same for the purposes of intent resolution (filtering). That is, if their action, data, type, class, and categories are the same. This does *not* compare any extra data included in the intents.

#### Parameters

*other*    The other Intent to compare against.

#### Returns

Returns true if action, data, type, class, and categories are the same.

public int **filterHashCode** ()

Added in [API level 1](#)

Generate hash code that matches semantics of filterEquals().

#### Returns

Returns the hash value of the action, data, type, class, and categories.

#### See Also

[filterEquals\(Intent\)](#)

public [String](#) **getAction** ()

Added in [API level 1](#)

Retrieve the general action to be performed, such as [ACTION\\_VIEW](#) ([/reference/android/content/Intent.html#ACTION\\_VIEW](#)). The action describes the general way the rest of the information in the intent should be interpreted – most importantly, what to do with the data returned by [getData\(\)](#) ([/reference/android/content/Intent.html#getData\(\)](#)).

#### Returns

The action of this intent or null if none is specified.

#### See Also

[setAction\(String\)](#)

public boolean[] **getBooleanArrayExtra** ([String](#) name)

Added in [API level 1](#)

Retrieve extended data from the intent.

#### Parameters

*name*    The name of the desired item.

#### Returns

the value of an item that previously added with putExtra() or null if no boolean array value was found.

#### See Also

[putExtra\(String, boolean\[\]\)](#)

public boolean **getBooleanExtra** ([String](#) name, boolean defaultValue)

Added in [API level 1](#)

Retrieve extended data from the intent.

#### Parameters

*name*    The name of the desired item.

*defaultValue*    the value to be returned if no value of the desired type is stored with the given name.

**Returns**

the value of an item that previously added with `putExtra()` or the default value if none was found.

**See Also**

[`putExtra\(String, boolean\)`](#)

public **Bundle** **getBundleExtra** ([`String`](#) name)

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

*name*     The name of the desired item.

**Returns**

the value of an item that previously added with `putExtra()` or null if no Bundle value was found.

**See Also**

[`putExtra\(String, Bundle\)`](#)

public byte[] **getByteArrayExtra** ([`String`](#) name)

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

*name*     The name of the desired item.

**Returns**

the value of an item that previously added with `putExtra()` or null if no byte array value was found.

**See Also**

[`putExtra\(String, byte\[\]\)`](#)

public byte **getByteExtra** ([`String`](#) name, byte defaultValue)

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

*name*     The name of the desired item.  
*defaultValue*     the value to be returned if no value of the desired type is stored with the given name.

**Returns**

the value of an item that previously added with `putExtra()` or the default value if none was found.

**See Also**

[`putExtra\(String, byte\)`](#)

public [`Set<String>`](#) **getCategories** ()

Added in [API level 1](#)

Return the set of all categories in the intent. If there are no categories, returns NULL.

**Returns**

The set of categories you can examine. Do not modify!

**See Also**

[`hasCategory\(String\)`](#)

[`addCategory\(String\)`](#)

public char[] **getCharArrayExtra** ([`String`](#) name)

Retrieve extended data from the intent.

#### Parameters

*name* The name of the desired item.

#### Returns

the value of an item that previously added with `putExtra()` or null if no char array value was found.

#### See Also

[`putExtra\(String, char\[\]\)`](#)

public char **getCharExtra** ([`String`](#) name, char defaultValue) Added in [API level 1](#)

Retrieve extended data from the intent.

#### Parameters

*name* The name of the desired item.  
*defaultValue* the value to be returned if no value of the desired type is stored with the given name.

#### Returns

the value of an item that previously added with `putExtra()` or the default value if none was found.

#### See Also

[`putExtra\(String, char\)`](#)

public [`CharSequence\[\]`](#) **getCharSequenceArrayExtra** ([`String`](#) name) Added in [API level 8](#)

Retrieve extended data from the intent.

#### Parameters

*name* The name of the desired item.

#### Returns

the value of an item that previously added with `putExtra()` or null if no `CharSequence` array value was found.

#### See Also

[`putExtra\(String, CharSequence\[\]\)`](#)

public [`ArrayList<CharSequence>`](#) **getCharSequenceArrayListExtra** ([`String`](#) name) Added in [API level 8](#)

Retrieve extended data from the intent.

#### Parameters

*name* The name of the desired item.

#### Returns

the value of an item that previously added with `putExtra()` or null if no `ArrayList` value was found.

#### See Also

[`putCharSequenceArrayListExtra\(String, ArrayList\)`](#)

public [`CharSequence`](#) **getCharSequenceExtra** ([`String`](#) name) Added in [API level 1](#)

Retrieve extended data from the intent.

#### Parameters

*name* The name of the desired item.

**Returns**

the value of an item that previously added with `putExtra()` or null if no CharSequence value was found.

**See Also**

[`putExtra\(String, CharSequence\)`](#)

public [`ClipData`](#) **getClipData** ()

Added in [API level 16](#)

Return the [`ClipData`](#) ([//reference/android/content/ClipData.html](#)) associated with this Intent. If there is none, returns null. See [`setClipData\(ClipData\)`](#) ([//reference/android/content/Intent.html#setClipData\(android.content.ClipData\)](#)) for more information.

**See Also**

[`setClipData\(ClipData\)`](#)

public [`ComponentName`](#) **getComponent** ()

Added in [API level 1](#)

Retrieve the concrete component associated with the intent. When receiving an intent, this is the component that was found to best handle it (that is, yourself) and will always be non-null; in all other cases it will be null unless explicitly set.

**Returns**

The name of the application component to handle the intent.

**See Also**

[`resolveActivity\(PackageManager\)`](#)

[`setComponent\(ComponentName\)`](#)

public [`Uri`](#) **getData** ()

Added in [API level 1](#)

Retrieve data this intent is operating on. This URI specifies the name of the data; often it uses the content: scheme, specifying data in a content provider. Other schemes may be handled by specific activities, such as http: by the web browser.

**Returns**

The URI of the data this intent is targeting or null.

**See Also**

[`getScheme\(\)`](#)

[`setData\(Uri\)`](#)

public [`String`](#) **getDataString** ()

Added in [API level 1](#)

The same as [`getData\(\)`](#) ([//reference/android/content/Intent.html#getData\(\)](#)), but returns the URI as an encoded String.

public double[] **getDoubleArrayExtra** ([`String`](#) name)

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

*name* The name of the desired item.

**Returns**

the value of an item that previously added with `putExtra()` or null if no double array value was found.

**See Also**

[`putExtra\(String, double\[\]\)`](#)

public double **getDoubleExtra** ([`String`](#) name, double defaultValue)



Retrieve extended data from the intent.

**Parameters**

<i>name</i>	The name of the desired item.
<i>defaultValue</i>	the value to be returned if no value of the desired type is stored with the given name.

**Returns**

the value of an item that previously added with `putExtra()` or the default value if none was found.

**See Also**

[`putExtra\(String, double\)`](#)

public **Bundle** **getExtras** ()

Added in [API level 1](#)

Retrieves a map of extended data from the intent.

**Returns**

the map of all extras previously added with `putExtra()`, or null if none have been added.

public int **getFlags** ()

Added in [API level 1](#)

Retrieve any special flags associated with this intent. You will normally just set them with [`setFlags\(int\)`](#) ([/reference/android/content/Intent.html#setFlags\(int\)](#)) and let the system take the appropriate action with them.

**Returns**

int The currently set flags.

**See Also**

[`setFlags\(int\)`](#)

public float[] **getFloatArrayExtra** ([String](#) name)

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

<i>name</i>	The name of the desired item.
-------------	-------------------------------

**Returns**

the value of an item that previously added with `putExtra()` or null if no float array value was found.

**See Also**

[`putExtra\(String, float\[\]\)`](#)

public float **getFloatExtra** ([String](#) name, float defaultValue)

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

<i>name</i>	The name of the desired item.
<i>defaultValue</i>	the value to be returned if no value of the desired type is stored with the given name.

**Returns**

the value of an item that previously added with `putExtra()`, or the default value if no such item is present

**See Also**

[`putExtra\(String, float\)`](#)

```
public int[] getIntArrayExtra (String name)
```

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

*name*     The name of the desired item.

**Returns**

the value of an item that previously added with `putExtra()` or null if no int array value was found.

**See Also**

[putExtra\(String, int\[\]\)](#)

```
public int getIntExtra (String name, int defaultValue)
```

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

*name*     The name of the desired item.  
*defaultValue*     the value to be returned if no value of the desired type is stored with the given name.

**Returns**

the value of an item that previously added with `putExtra()` or the default value if none was found.

**See Also**

[putExtra\(String, int\)](#)

```
public ArrayList<Integer> getIntegerArrayListExtra (String name)
```

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

*name*     The name of the desired item.

**Returns**

the value of an item that previously added with `putExtra()` or null if no `ArrayList` value was found.

**See Also**

[putIntegerArrayListExtra\(String, ArrayList\)](#)

```
public static Intent getIntent (String uri)
```

Added in [API level 1](#)

This method was deprecated in API level 4.

Use [parseUri\(String, int\)](#) ([`/reference/android/content/Intent.html#parseUri\(java.lang.String, int\)`](/reference/android/content/Intent.html#parseUri(java.lang.String, int))) instead.

Call [parseUri\(String, int\)](#) ([`/reference/android/content/Intent.html#parseUri\(java.lang.String, int\)`](/reference/android/content/Intent.html#parseUri(java.lang.String, int))) with 0 flags.

**Throws**

[URISyntaxException](#)

```
public static Intent getIntentOld (String uri)
```

Added in [API level 1](#)**Throws**

[URISyntaxException](#)

public long[] **getLongArrayExtra** ([String](#) name)

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

*name*     The name of the desired item.

**Returns**

the value of an item that previously added with `putExtra()` or null if no long array value was found.

**See Also**

[putExtra\(String, long\[\]\)](#)

public long **getLongExtra** ([String](#) name, long defaultValue)

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

*name*     The name of the desired item.

*defaultValue*     the value to be returned if no value of the desired type is stored with the given name.

**Returns**

the value of an item that previously added with `putExtra()` or the default value if none was found.

**See Also**

[putExtra\(String, long\)](#)

public [String](#) **getPackage** ()

Added in [API level 4](#)

Retrieve the application package name this Intent is limited to. When resolving an Intent, if non-null this limits the resolution to only components in the given application package.

**Returns**

The name of the application package for the Intent.

**See Also**

[resolveActivity\(PackageManager\)](#)

[setPackage\(String\)](#)

public [Parcelable](#)[] **getParcelableArrayExtra** ([String](#) name)

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

*name*     The name of the desired item.

**Returns**

the value of an item that previously added with `putExtra()` or null if no `Parcelable[]` value was found.

**See Also**

[putExtra\(String, Parcelable\[\]\)](#)

public [ArrayList](#)<[T](#)> **getParcelableArrayListExtra** ([String](#) name)

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

*name*     The name of the desired item.

**Returns**

the value of an item that previously added with `putExtra()` or null if no `ArrayList` value was found.

**See Also**

[`putParcelableArrayListExtra\(String, ArrayList\)`](#)

public T **getParcelableExtra** (String name)

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

*name*    The name of the desired item.

**Returns**

the value of an item that previously added with `putExtra()` or null if no `Parcelable` value was found.

**See Also**

[`putExtra\(String, Parcelable\)`](#)

public String **getScheme** ()

Added in [API level 1](#)

Return the scheme portion of the intent's data. If the data is null or does not include a scheme, null is returned. Otherwise, the scheme prefix without the final ':' is returned, i.e. "http".

This is the same as calling `getData().getScheme()` (and checking for null data).

**Returns**

The scheme of this intent.

**See Also**

[`getData\(\)`](#)

public Intent **getSelector** ()

Added in [API level 15](#)

Return the specific selector associated with this Intent. If there is none, returns null. See

[`setSelector\(Intent\)`](#) ([/reference/android/content/Intent.html#setSelector\(android.content.Intent\)](#)) for more information.

**See Also**

[`setSelector\(Intent\)`](#)

public Serializable **getSerializableExtra** (String name)

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

*name*    The name of the desired item.

**Returns**

the value of an item that previously added with `putExtra()` or null if no `Serializable` value was found.

**See Also**

[`putExtra\(String, Serializable\)`](#)

public short[] **getShortArrayExtra** (String name)

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

*name*    The name of the desired item.

**Returns**

the value of an item that previously added with `putExtra()` or null if no short array value was found.

**See Also**

[`putExtra\(String, short\[\]\)`](#)

public short **`getShortExtra`** ([`String`](#) name, short defaultValue)

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

<i>name</i>	The name of the desired item.
<i>defaultValue</i>	the value to be returned if no value of the desired type is stored with the given name.

**Returns**

the value of an item that previously added with `putExtra()` or the default value if none was found.

**See Also**

[`putExtra\(String, short\)`](#)

public [`Rect`](#) **`getSourceBounds`** ()

Added in [API level 7](#)

Get the bounds of the sender of this intent, in screen coordinates. This can be used as a hint to the receiver for animations and the like. Null means that there is no source bounds.

public [`String\[\]`](#) **`getStringArrayExtra`** ([`String`](#) name)

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

<i>name</i>	The name of the desired item.
-------------	-------------------------------

**Returns**

the value of an item that previously added with `putExtra()` or null if no String array value was found.

**See Also**

[`putExtra\(String, String\[\]\)`](#)

public [`ArrayList<String>`](#) **`getStringArrayListExtra`** ([`String`](#) name)

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

<i>name</i>	The name of the desired item.
-------------	-------------------------------

**Returns**

the value of an item that previously added with `putExtra()` or null if no ArrayList value was found.

**See Also**

[`putStringArrayListExtra\(String, ArrayList\)`](#)

public [`String`](#) **`getStringExtra`** ([`String`](#) name)

Added in [API level 1](#)

Retrieve extended data from the intent.

**Parameters**

<i>name</i>	The name of the desired item.
-------------	-------------------------------

**Returns**

the value of an item that previously added with `putExtra()` or null if no String value was found.

**See Also**

[`putExtra\(String, String\)`](#)

public **String** **getType** ()

Added in [API level 1](#)

Retrieve any explicit MIME type included in the intent. This is usually null, as the type is determined by the intent data.

**Returns**

If a type was manually set, it is returned; else null is returned.

**See Also**

[`resolveType\(ContentResolver\)`](#)

[`setType\(String\)`](#)

public boolean **hasCategory** ([String](#) category)

Added in [API level 1](#)

Check if a category exists in the intent.

**Parameters**

*category*    The category to check.

**Returns**

boolean True if the intent contains the category, else false.

**See Also**

[`getCategories\(\)`](#)

[`addCategory\(String\)`](#)

public boolean **hasExtra** ([String](#) name)

Added in [API level 1](#)

Returns true if an extra value is associated with the given name.

**Parameters**

*name*    the extra's name

**Returns**

true if the given extra is present.

public boolean **hasFileDescriptors** ()

Added in [API level 1](#)

Returns true if the Intent's extras contain a parcelled file descriptor.

**Returns**

true if the Intent contains a parcelled file descriptor.

public static **Intent** **makeMainActivity** ([ComponentName](#) mainActivity)

Added in [API level 11](#)

Create an intent to launch the main (root) activity of a task. This is the Intent that is started when the application's is launched from Home. For anything else that wants to launch an application in the same way, it is important that they use an Intent structured the same way, and can use this function to ensure this is the case.

The returned Intent has the given Activity component as its explicit component, [ACTION\\_MAIN](#) ([//reference/android/content/Intent.html#ACTION\\_MAIN](#)) as its action, and includes the category [CATEGORY\\_LAUNCHER](#) ([//reference/android/content/Intent.html#CATEGORY\\_LAUNCHER](#)). This does *not* have [FLAG\\_ACTIVITY\\_NEW\\_TASK](#) ([//reference/android/content/Intent.html#FLAG\\_ACTIVITY\\_NEW\\_TASK](#)) set, though typically you will

want to do that through [`addFlags\(int\)`](#) ([/reference/android/content/Intent.html#addFlags\(int\)](#)) on the returned Intent.

#### Parameters

*mainActivity*    The main activity component that this Intent will launch.

#### Returns

Returns a newly created Intent that can be used to launch the activity as a main application entry.

#### See Also

[`setClass\(Context, Class\)`](#)  
[`setComponent\(ComponentName\)`](#)

public static [`Intent`](#) **makeMainSelectorActivity** ([String](#) selectorAction, [String](#) selectorCategory) Added in [API level 15](#)

Make an Intent for the main activity of an application, without specifying a specific activity to run but giving a selector to find the activity. This results in a final Intent that is structured the same as when the application is launched from Home. For anything else that wants to launch an application in the same way, it is important that they use an Intent structured the same way, and can use this function to ensure this is the case.

The returned Intent has [ACTION\\_MAIN](#) ([/reference/android/content/Intent.html#ACTION\\_MAIN](#)) as its action, and includes the category [CATEGORY\\_LAUNCHER](#) ([/reference/android/content/Intent.html#CATEGORY\\_LAUNCHER](#)). This does *not* have [FLAG\\_ACTIVITY\\_NEW\\_TASK](#) ([/reference/android/content/Intent.html#FLAG\\_ACTIVITY\\_NEW\\_TASK](#)) set, though typically you will want to do that through [`addFlags\(int\)`](#) ([/reference/android/content/Intent.html#addFlags\(int\)](#)) on the returned Intent.

#### Parameters

*selectorAction*    The action name of the Intent's selector.  
*selectorCategory*    The name of a category to add to the Intent's selector.

#### Returns

Returns a newly created Intent that can be used to launch the activity as a main application entry.

#### See Also

[`setSelector\(Intent\)`](#)

public static [`Intent`](#) **makeRestartActivityTask** ([ComponentName](#) mainActivity) Added in [API level 11](#)

Make an Intent that can be used to re-launch an application's task in its base state. This is like [`makeMainActivity\(ComponentName\)`](#) ([/reference/android/content/Intent.html#makeMainActivity\(android.content.ComponentName\)](#)), but also sets the flags [FLAG\\_ACTIVITY\\_NEW\\_TASK](#) ([/reference/android/content/Intent.html#FLAG\\_ACTIVITY\\_NEW\\_TASK](#)) and [FLAG\\_ACTIVITY\\_CLEAR\\_TASK](#) ([/reference/android/content/Intent.html#FLAG\\_ACTIVITY\\_CLEAR\\_TASK](#)).

#### Parameters

*mainActivity*    The activity component that is the root of the task; this is the activity that has been published in the application's manifest as the main launcher icon.

#### Returns

Returns a newly created Intent that can be used to relaunch the activity's task in its root state.

public static String **normalizeMimeType** (String type)

Added in API level 16

Normalize a MIME data type.

A normalized MIME type has white-space trimmed, content-type parameters removed, and is lower-case. This aligns the type with Android best practices for intent filtering.

For example, "text/plain; charset=utf-8" becomes "text/plain". "text/x-vCard" becomes "text/x-vcard".

All MIME types received from outside Android (such as user input, or external sources like Bluetooth, NFC, or the Internet) should be normalized before they are used to create an Intent.

#### Parameters

*type* MIME data type to normalize

#### Returns

normalized MIME data type, or null if the input was null

#### See Also

setType(String)

setTypeAndNormalize(String)

public static Intent **parseIntent** (Resources resources, XmlPullParser parser, AttributeSet attrs)

Added in API level 1

Parses the "intent" element (and its children) from XML and instantiates an Intent object. The given XML parser should be located at the tag where parsing should start (often named "intent"), from which the basic action, data, type, and package and class name will be retrieved. The function will then parse in to any child elements, looking for tags to add categories and to attach extra data to the intent.

#### Parameters

*resources* The Resources to use when inflating resources.

*parser* The XML parser pointing at an "intent" tag.

*attrs* The AttributeSet interface for retrieving extended attribute data at the current *parser* location.

#### Returns

An Intent object matching the XML data.

#### Throws

XmlPullParserException If there was an XML parsing error.

IOException If there was an I/O error.

public static Intent **parseUri** (String uri, int flags)

Added in API level 4

Create an intent from a URI. This URI may encode the action, category, and other intent fields, if it was returned by toUri(int) ([toUri\(int\)](http://reference.android.com/content/Intent.html#toUri(int))). If the Intent was not generate by toUri(), its data will be the entire URI and its action will be ACTION\_VIEW.

The URI given here must not be relative -- that is, it must include the scheme and full path.

#### Parameters

*uri* The URI to turn into an Intent.

*flags* Additional processing flags. Either 0 or URI\_INTENT\_SCHEME.

#### Returns

Intent The newly created Intent object.

#### Throws



[URISyntaxException](#) Throws `URISyntaxError` if the basic URI syntax is bad (as parsed by the `Uri` class) or the Intent data within the URI is invalid.

**See Also**

[toUri\(int\)](#)

public [Intent](#) **putCharSequenceArrayListExtra** ([String](#) name, [ArrayList<CharSequence>](#) value)

Added in [API level 8](#)

Add extended data to the intent. The name must include a package prefix, for example the app `com.android.contacts` would use names like `"com.android.contacts.ShowAll"`.

**Parameters**

*name* The name of the extra data, with package prefix.  
*value* The `ArrayList` data value.

**Returns**

Returns the same `Intent` object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getCharSequenceArrayListExtra\(String\)](#)

public [Intent](#) **putExtra** ([String](#) name, `double[]` value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app `com.android.contacts` would use names like `"com.android.contacts.ShowAll"`.

**Parameters**

*name* The name of the extra data, with package prefix.  
*value* The double array data value.

**Returns**

Returns the same `Intent` object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getDoubleArrayExtra\(String\)](#)

public [Intent](#) **putExtra** ([String](#) name, `int` value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app `com.android.contacts` would use names like `"com.android.contacts.ShowAll"`.

**Parameters**

*name* The name of the extra data, with package prefix.  
*value* The integer data value.

**Returns**

Returns the same `Intent` object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getIntExtra\(String, int\)](#)

public [Intent](#) **putExtra** ([String](#) name, [CharSequence](#) value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the

app com.android.contacts would use names like "com.android.contacts.ShowAll".

**Parameters**

*name*    The name of the extra data, with package prefix.  
*value*    The CharSequence data value.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getCharSequenceExtra\(String\)](#)

public [Intent](#) **putExtra** ([String](#) name, char value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

**Parameters**

*name*    The name of the extra data, with package prefix.  
*value*    The char data value.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getCharExtra\(String, char\)](#)

public [Intent](#) **putExtra** ([String](#) name, [Bundle](#) value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

**Parameters**

*name*    The name of the extra data, with package prefix.  
*value*    The Bundle data value.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getBundleExtra\(String\)](#)

public [Intent](#) **putExtra** ([String](#) name, [Parcelable\[\]](#) value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

**Parameters**

*name*    The name of the extra data, with package prefix.  
*value*    The Parcelable[] data value.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)

[removeExtra\(String\)](#)  
[getParcelableArrayExtra\(String\)](#)

public [Intent](#) **putExtra** ([String](#) name, [Serializable](#) value) Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

#### Parameters

*name*    The name of the extra data, with package prefix.  
*value*    The Serializable data value.

#### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

#### See Also

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getSerializableExtra\(String\)](#)

public [Intent](#) **putExtra** ([String](#) name, int[] value) Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

#### Parameters

*name*    The name of the extra data, with package prefix.  
*value*    The int array data value.

#### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

#### See Also

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getIntArrayExtra\(String\)](#)

public [Intent](#) **putExtra** ([String](#) name, float value) Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

#### Parameters

*name*    The name of the extra data, with package prefix.  
*value*    The float data value.

#### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

#### See Also

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getFloatExtra\(String, float\)](#)

public [Intent](#) **putExtra** ([String](#) name, byte[] value) Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

#### Parameters

*name*    The name of the extra data, with package prefix.

*value* The byte array data value.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)

[removeExtra\(String\)](#)

[getByteArrayExtra\(String\)](#)

public [Intent](#) **putExtra** (String name, long[] value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

**Parameters**

*name* The name of the extra data, with package prefix.

*value* The byte array data value.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)

[removeExtra\(String\)](#)

[getLongArrayExtra\(String\)](#)

public [Intent](#) **putExtra** (String name, [Parcelable](#) value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

**Parameters**

*name* The name of the extra data, with package prefix.

*value* The Parcelable data value.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)

[removeExtra\(String\)](#)

[getParcelableExtra\(String\)](#)

public [Intent](#) **putExtra** (String name, float[] value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

**Parameters**

*name* The name of the extra data, with package prefix.

*value* The float array data value.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)

[removeExtra\(String\)](#)

[getFloatArrayExtra\(String\)](#)

public [Intent](#) **putExtra** ([String](#) name, long value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

#### Parameters

*name*    The name of the extra data, with package prefix.  
*value*    The long data value.

#### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

#### See Also

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getLongExtra\(String, long\)](#)

public [Intent](#) **putExtra** ([String](#) name, [String\[\]](#) value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

#### Parameters

*name*    The name of the extra data, with package prefix.  
*value*    The String array data value.

#### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

#### See Also

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getStringArrayExtra\(String\)](#)

public [Intent](#) **putExtra** ([String](#) name, boolean value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

#### Parameters

*name*    The name of the extra data, with package prefix.  
*value*    The boolean data value.

#### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

#### See Also

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getBooleanExtra\(String, boolean\)](#)

public [Intent](#) **putExtra** ([String](#) name, [boolean\[\]](#) value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

#### Parameters

*name*    The name of the extra data, with package prefix.  
*value*    The boolean array data value.

#### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getBooleanArrayExtra\(String\)](#)

public [Intent](#) **putExtra** ([String](#) name, short value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

**Parameters**

*name*    The name of the extra data, with package prefix.  
*value*    The short data value.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getShortExtra\(String, short\)](#)

public [Intent](#) **putExtra** ([String](#) name, double value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

**Parameters**

*name*    The name of the extra data, with package prefix.  
*value*    The double data value.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getDoubleExtra\(String, double\)](#)

public [Intent](#) **putExtra** ([String](#) name, short[] value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

**Parameters**

*name*    The name of the extra data, with package prefix.  
*value*    The short array data value.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getShortArrayExtra\(String\)](#)

public [Intent](#) **putExtra** ([String](#) name, [String](#) value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

**Parameters**

*name* The name of the extra data, with package prefix.  
*value* The String data value.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getStringExtra\(String\)](#)

public [Intent](#) **putExtra** (String name, byte value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

**Parameters**

*name* The name of the extra data, with package prefix.  
*value* The byte data value.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getBytesExtra\(String, byte\)](#)

public [Intent](#) **putExtra** (String name, char[] value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

**Parameters**

*name* The name of the extra data, with package prefix.  
*value* The char array data value.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getCharArrayExtra\(String\)](#)

public [Intent](#) **putExtra** (String name, [CharSequence\[\]](#) value)

Added in [API level 8](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

**Parameters**

*name* The name of the extra data, with package prefix.  
*value* The CharSequence array data value.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[putExtras\(Intent\)](#)  
[removeExtra\(String\)](#)  
[getCharSequenceArrayExtra\(String\)](#)

public [Intent](#) **putExtras** ([Intent](#) src)

Added in [API level 1](#)

Copy all extras in 'src' in to this intent.

#### Parameters

*src*     Contains the extras to copy.

#### See Also

[putExtra\(String, Bundle\)](#)

public [Intent](#) **putExtras** ([Bundle](#) extras)

Added in [API level 1](#)

Add a set of extended data to the intent. The keys must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

#### Parameters

*extras*     The [Bundle](#) of extras to add to this intent.

#### See Also

[putExtra\(String, Bundle\)](#)

[removeExtra\(String\)](#)

public [Intent](#) **putIntegerArrayListExtra** ([String](#) name, [ArrayList](#)<[Integer](#)> value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

#### Parameters

*name*     The name of the extra data, with package prefix.

*value*     The [ArrayList](#) data value.

#### Returns

Returns the same [Intent](#) object, for chaining multiple calls into a single statement.

#### See Also

[putExtras\(Intent\)](#)

[removeExtra\(String\)](#)

[getIntegerArrayListExtra\(String\)](#)

public [Intent](#) **putParcelableArrayListExtra** ([String](#) name, [ArrayList](#)<? extends [Parcelable](#)> value)

Added in [API level 1](#)

Add extended data to the intent. The name must include a package prefix, for example the app com.android.contacts would use names like "com.android.contacts.ShowAll".

#### Parameters

*name*     The name of the extra data, with package prefix.

*value*     The [ArrayList](#) data value.

#### Returns

Returns the same [Intent](#) object, for chaining multiple calls into a single statement.

#### See Also

[putExtras\(Intent\)](#)

[removeExtra\(String\)](#)

[getParcelableArrayListExtra\(String\)](#)

public [Intent](#) **putStringArrayListExtra** ([String](#) name, [ArrayList](#)<[String](#)> value)

Added in [API level 1](#)



Add extended data to the intent. The name must include a package prefix, for example the app `com.android.contacts` would use names like `"com.android.contacts.ShowAll"`.

#### Parameters

*name* The name of the extra data, with package prefix.  
*value* The ArrayList data value.

#### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

#### See Also

[`putExtras\(Intent\)`](#)  
[`removeExtra\(String\)`](#)  
[`getStringArrayListExtra\(String\)`](#)

public void **readFromParcel** ([`Parcel`](#) in)

Added in [`API level 1`](#)

public void **removeCategory** ([`String`](#) category)

Added in [`API level 1`](#)

Remove a category from an intent.

#### Parameters

*category* The category to remove.

#### See Also

[`addCategory\(String\)`](#)

public void **removeExtra** ([`String`](#) name)

Added in [`API level 1`](#)

Remove extended data from the intent.

#### See Also

[`putExtra\(String, Bundle\)`](#)

public [`Intent`](#) **replaceExtras** ([`Bundle`](#) extras)

Added in [`API level 3`](#)

Completely replace the extras in the Intent with the given Bundle of extras.

#### Parameters

*extras* The new set of extras in the Intent, or null to erase all extras.

public [`Intent`](#) **replaceExtras** ([`Intent`](#) src)

Added in [`API level 3`](#)

Completely replace the extras in the Intent with the extras in the given Intent.

#### Parameters

*src* The exact extras contained in this Intent are copied into the target intent, replacing any that were previously there.

public [`ComponentName`](#) **resolveActivity** ([`PackageManager`](#) pm)

Added in [`API level 1`](#)

Return the Activity component that should be used to handle this intent. The appropriate component is determined based on the information in the intent, evaluated as follows:

If [`getComponent\(\)`](#) ([`//reference/android/content/Intent.html#getComponent\(\)`](#)) returns an explicit class, that is returned without any further consideration.

The activity must handle the [`CATEGORY\_DEFAULT`](#) ([`//reference/android/content/Intent.html#CATEGORY\_DEFAULT`](#)) Intent category to be considered.

If [`getAction\(\)`](#) ([`//reference/android/content/Intent.html#getAction\(\)`](#)) is non-NULL, the

activity must handle this action.

If [`resolveType\(ContentResolver\)`](#) ([/reference/android/content/Intent.html#resolveType\(android.content.ContentResolver\)](#)) returns non-NULL, the activity must handle this type.

If [`addCategory\(String\)`](#) ([/reference/android/content/Intent.html#addCategory\(java.lang.String\)](#)) has added any categories, the activity must handle ALL of the categories specified.

If [`getPackage\(\)`](#) ([/reference/android/content/Intent.html#getPackage\(\)](#)) is non-NULL, only activity components in that application package will be considered.

If there are no activities that satisfy all of these conditions, a null string is returned.

If multiple activities are found to satisfy the intent, the one with the highest priority will be used. If there are multiple activities with the same priority, the system will either pick the best activity based on user preference, or resolve to a system class that will allow the user to pick an activity and forward from there.

This method is implemented simply by calling [`resolveActivity\(Intent, int\)`](#) ([/reference/android/content/pm/PackageManager.html#resolveActivity\(android.content.Intent, int\)](#)) with the "defaultOnly" parameter true.

This API is called for you as part of starting an activity from an intent. You do not normally need to call it yourself.

#### Parameters

*pm* The package manager with which to resolve the Intent.

#### Returns

Name of the component implementing an activity that can display the intent.

#### See Also

[`setComponent\(ComponentName\)`](#)

[`getComponent\(\)`](#)

[`resolveActivityInfo\(PackageManager, int\)`](#)

**public [`ActivityInfo resolveActivityInfo`](#) ([`PackageManager pm`](#), [`int flags`](#))** Added in [API level 1](#)

Resolve the Intent into an [`ActivityInfo`](#) ([/reference/android/content/pm/ActivityInfo.html](#)) describing the activity that should execute the intent. Resolution follows the same rules as described for [`resolveActivity\(PackageManager\)`](#) ([/reference/android/content/Intent.html#resolveActivity\(android.content.pm.PackageManager\)](#)), but you get back the completely information about the resolved activity instead of just its class name.

#### Parameters

*pm* The package manager with which to resolve the Intent.

*flags* Addition information to retrieve as per [`PackageManager.getActivityInfo\(\)`](#).

#### Returns

[`PackageManager.ActivityInfo`](#)

#### See Also

[`resolveActivity\(PackageManager\)`](#)

**public [`String resolveType`](#) ([`ContentResolver resolver`](#))** Added in [API level 1](#)

Return the MIME data type of this intent. If the type field is explicitly set, that is simply returned. Otherwise, if the data is set, the type of that data is returned. If neither fields are

set, a null is returned.

#### Parameters

*resolver* A `ContentResolver` that can be used to determine the MIME type of the intent's data.

#### Returns

The MIME type of this intent.

#### See Also

[`getType\(\)`](#)

[`resolveType\(Context\)`](#)

public [`String`](#) **`resolveType`** ([`Context`](#) context)

Added in [`API level 1`](#)

Return the MIME data type of this intent. If the type field is explicitly set, that is simply returned. Otherwise, if the data is set, the type of that data is returned. If neither fields are set, a null is returned.

#### Returns

The MIME type of this intent.

#### See Also

[`getType\(\)`](#)

[`resolveType\(ContentResolver\)`](#)

public [`String`](#) **`resolveTypeIfNeeded`** ([`ContentResolver`](#) resolver)

Added in [`API level 1`](#)

Return the MIME data type of this intent, only if it will be needed for intent resolution. This is not generally useful for application code; it is used by the frameworks for communicating with back-end system services.

#### Parameters

*resolver* A `ContentResolver` that can be used to determine the MIME type of the intent's data.

#### Returns

The MIME type of this intent, or null if it is unknown or not needed.

public [`Intent`](#) **`setAction`** ([`String`](#) action)

Added in [`API level 1`](#)

Set the general action to be performed.

#### Parameters

*action* An action name, such as `ACTION_VIEW`. Application-specific actions should be prefixed with the vendor's package name.

#### Returns

Returns the same `Intent` object, for chaining multiple calls into a single statement.

#### See Also

[`getAction\(\)`](#)

public [`Intent`](#) **`setClass`** ([`Context`](#) packageContext, [`Class`](#)<?> cls)

Added in [`API level 1`](#)

Convenience for calling [`setComponent\(ComponentName\)`](#) ([`/reference/android/content/Intent.html#setComponent\(android.content.ComponentName\)`](#)) with the name returned by a [`Class`](#) ([`/reference/java/lang/Class.html`](#)) object.

#### Parameters

*packageContext* A `Context` of the application package implementing this class.

*cls*      The class name to set, equivalent to `setClassName(context, cls.getName())`.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[setComponent\(ComponentName\)](#)

public [Intent](#) **setClassName** ([Context](#) packageContext, [String](#) className)

Added in [API level 1](#)

Convenience for calling [setComponent\(ComponentName\)](#) ([/reference/android/content/Intent.html#setComponent\(android.content.ComponentName\)](#)) with an explicit class name.

**Parameters**

*packageContext*    A Context of the application package implementing this class.  
*className*          The name of a class inside of the application package that will be used as the component for this Intent.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[setComponent\(ComponentName\)](#)  
[setClass\(Context, Class\)](#)

public [Intent](#) **setClassName** ([String](#) packageName, [String](#) className)    Added in [API level 1](#)

Convenience for calling [setComponent\(ComponentName\)](#) ([/reference/android/content/Intent.html#setComponent\(android.content.ComponentName\)](#)) with an explicit application package name and class name.

**Parameters**

*packageName*      The name of the package implementing the desired component.  
*className*          The name of a class inside of the application package that will be used as the component for this Intent.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[setComponent\(ComponentName\)](#)  
[setClass\(Context, Class\)](#)

public void **setClipData** ([ClipData](#) clip)

Added in [API level 16](#)

Set a [ClipData](#) ([/reference/android/content/ClipData.html](#)) associated with this Intent. This replaces any previously set ClipData.

The ClipData in an intent is not used for Intent matching or other such operations. Semantically it is like extras, used to transmit additional data with the Intent. The main feature of using this over the extras for data is that [FLAG\\_GRANT\\_READ\\_URI\\_PERMISSION](#) ([/reference/android/content/Intent.html#FLAG\\_GRANT\\_READ\\_URI\\_PERMISSION](#)) and [FLAG\\_GRANT\\_WRITE\\_URI\\_PERMISSION](#) ([/reference/android/content/Intent.html#FLAG\\_GRANT\\_WRITE\\_URI\\_PERMISSION](#)) will operate on any URI items included in the clip data. This is useful, in particular, if you want to transmit an Intent containing multiple content : URIs for which the recipient may not have global permission to access the content provider.

If the ClipData contains items that are themselves Intents, any grant flags in those Intents will be ignored. Only the top-level flags of the main Intent are respected, and will be applied

to all Uri or Intent items in the clip (or sub-items of the clip).

The MIME type, label, and icon in the ClipData object are not directly used by Intent. Applications should generally rely on the MIME type of the Intent itself, not what it may find in the ClipData. A common practice is to construct a ClipData for use with an Intent with a MIME type of `"*/*"`.

#### Parameters

*clip* The new clip to set. May be null to clear the current clip.

public **Intent** **setComponent** (**ComponentName** component)

Added in [API level 1](#)

(Usually optional) Explicitly set the component to handle the intent. If left with the default value of null, the system will determine the appropriate class to use based on the other fields (action, data, type, categories) in the Intent. If this class is defined, the specified class will always be used regardless of the other fields. You should only set this value when you know you absolutely want a specific class to be used; otherwise it is better to let the system find the appropriate class so that you will respect the installed applications and user preferences.

#### Parameters

*component* The name of the application component to handle the intent, or null to let the system find one for you.

#### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

#### See Also

[`setClass\(Context, Class\)`](#)  
[`setClassName\(Context, String\)`](#)  
[`setClassName\(String, String\)`](#)  
[`getComponent\(\)`](#)  
[`resolveActivity\(PackageManager\)`](#)

public **Intent** **setData** (**Uri** data)

Added in [API level 1](#)

Set the data this intent is operating on. This method automatically clears any type that was previously set by [`setType\(String\)`](#) ([/reference/android/content/Intent.html#setType\(java.lang.String\)](#)) or [`setTypeAndNormalize\(String\)`](#) ([/reference/android/content/Intent.html#setTypeAndNormalize\(java.lang.String\)](#)).

*Note: scheme matching in the Android framework is case-sensitive, unlike the formal RFC. As a result, you should always write your Uri with a lower case scheme, or use [`normalizeScheme\(\)`](#) ([/reference/android/net/Uri.html#normalizeScheme\(\)](#)) or [`setDataAndNormalize\(Uri\)`](#) ([/reference/android/content/Intent.html#setDataAndNormalize\(android.net.Uri\)](#)) to ensure that the scheme is converted to lower case.*

#### Parameters

*data* The Uri of the data this intent is now targeting.

#### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

#### See Also

[`getData\(\)`](#)  
[`setDataAndNormalize\(Uri\)`](#)  
[`normalizeScheme\(\)`](#)

public **Intent** **setDataAndNormalize** (**Uri** data)

Added in [API level 16](#)

Normalize and set the data this intent is operating on.

This method automatically clears any type that was previously set (for example, by [`setType\(String\)`](#) ([/reference/android/content/Intent.html#setType\(java.lang.String\)](#))).

The data Uri is normalized using [`normalizeScheme\(\)`](#) ([/reference/android/net/Uri.html#normalizeScheme\(\)](#)) before it is set, so really this is just a convenience method for

```
setData(data.normalize())
```

#### Parameters

*data*    The Uri of the data this intent is now targeting.

#### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

#### See Also

[`getData\(\)`](#)

[`setType\(String\)`](#)

[`normalizeScheme\(\)`](#)

**public Intent setDataAndType** ([Uri](#) data, [String](#) type)

Added in [API level 1](#)

(Usually optional) Set the data for the intent along with an explicit MIME data type. This method should very rarely be used -- it allows you to override the MIME type that would ordinarily be inferred from the data with your own type given here.

*Note: MIME type and Uri scheme matching in the Android framework is case-sensitive, unlike the formal RFC definitions. As a result, you should always write these elements with lower case letters, or use [`normalizeMimeType\(String\)`](#) ([/reference/android/content/Intent.html#normalizeMimeType\(java.lang.String\)](#)) or [`normalizeScheme\(\)`](#) ([/reference/android/net/Uri.html#normalizeScheme\(\)](#)) or [`setDataAndTypeAndNormalize\(Uri, String\)`](#) ([/reference/android/content/Intent.html#setDataAndTypeAndNormalize\(android.net.Uri, java.lang.String\)](#)) to ensure that they are converted to lower case.*

#### Parameters

*data*    The Uri of the data this intent is now targeting.

*type*    The MIME type of the data being handled by this intent.

#### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

#### See Also

[`setType\(String\)`](#)

[`setData\(Uri\)`](#)

[`normalizeMimeType\(String\)`](#)

[`normalizeScheme\(\)`](#)

[`setDataAndTypeAndNormalize\(Uri, String\)`](#)

**public Intent setDataAndTypeAndNormalize** ([Uri](#) data, [String](#) type)

Added in [API level 16](#)

(Usually optional) Normalize and set both the data Uri and an explicit MIME data type. This method should very rarely be used -- it allows you to override the MIME type that would ordinarily be inferred from the data with your own type given here.

The data Uri and the MIME type are normalize using [`normalizeScheme\(\)`](#) ([/reference/android/net/Uri.html#normalizeScheme\(\)](#)) and [`normalizeMimeType\(String\)`](#) ([/reference/android/content/Intent.html#normalizeMimeType\(java.lang.String\)](#)) before they are set, so really this is just a convenience method for

```
setDataAndType(data.normalize(), Intent.normalizeMimeType(type))
```

**Parameters**

*data*    The Uri of the data this intent is now targeting.

*type*    The MIME type of the data being handled by this intent.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[setType\(String\)](#)  
[setData\(Uri\)](#)  
[setDataAndType\(Uri, String\)](#)  
[normalizeMimeType\(String\)](#)  
[normalizeScheme\(\)](#)

public void **setExtrasClassLoader** ([ClassLoader](#) loader) Added in [API level 1](#)

Sets the ClassLoader that will be used when unmarshalling any Parcelable values from the extras of this Intent.

**Parameters**

*loader*    a ClassLoader, or null to use the default loader at the time of unmarshalling.

public [Intent](#) **setFlags** (int flags) Added in [API level 1](#)

Set special flags controlling how this intent is handled. Most values here depend on the type of component being executed by the Intent, specifically the FLAG\_ACTIVITY\_\* flags are all for use with [Context.startActivity\(\)](#) ([//reference/android/content/Context.html#startActivity\(android.content.Intent\)](#)) and the FLAG\_RECEIVER\_\* flags are all for use with [Context.sendBroadcast\(\)](#) ([//reference/android/content/Context.html#sendBroadcast\(android.content.Intent\)](#)).

See the [Tasks and Back Stack](#) ([//guide/topics/fundamentals/tasks-and-back-stack.html](#)) documentation for important information on how some of these options impact the behavior of your application.

**Parameters**

*flags*    The desired flags.

**Returns**

Returns the same Intent object, for chaining multiple calls into a single statement.

**See Also**

[getFlags\(\)](#)  
[addFlags\(int\)](#)  
[FLAG\\_GRANT\\_READ\\_URI\\_PERMISSION](#)  
[FLAG\\_GRANT\\_WRITE\\_URI\\_PERMISSION](#)  
[FLAG\\_DEBUG\\_LOG\\_RESOLUTION](#)  
[FLAG\\_FROM\\_BACKGROUND](#)  
[FLAG\\_ACTIVITY\\_BROUGHT\\_TO\\_FRONT](#)  
[FLAG\\_ACTIVITY\\_CLEAR\\_TASK](#)  
[FLAG\\_ACTIVITY\\_CLEAR\\_TOP](#)  
[FLAG\\_ACTIVITY\\_CLEAR\\_WHEN\\_TASK\\_RESET](#)  
[FLAG\\_ACTIVITY\\_EXCLUDE\\_FROM\\_RECENTS](#)  
[FLAG\\_ACTIVITY\\_FORWARD\\_RESULT](#)  
[FLAG\\_ACTIVITY\\_LAUNCHED\\_FROM\\_HISTORY](#)  
[FLAG\\_ACTIVITY\\_MULTIPLE\\_TASK](#)

[FLAG\\_ACTIVITY\\_NEW\\_TASK](#)  
[FLAG\\_ACTIVITY\\_NO\\_ANIMATION](#)  
[FLAG\\_ACTIVITY\\_NO\\_HISTORY](#)  
[FLAG\\_ACTIVITY\\_NO\\_USER\\_ACTION](#)  
[FLAG\\_ACTIVITY\\_PREVIOUS\\_IS\\_TOP](#)  
[FLAG\\_ACTIVITY\\_RESET\\_TASK\\_IF\\_NEEDED](#)  
[FLAG\\_ACTIVITY\\_REORDER\\_TO\\_FRONT](#)  
[FLAG\\_ACTIVITY\\_SINGLE\\_TOP](#)  
[FLAG\\_ACTIVITY\\_TASK\\_ON\\_HOME](#)  
[FLAG\\_RECEIVER\\_REGISTERED\\_ONLY](#)

public [Intent](#) **setPackage** ([String](#) packageName)

Added in [API level 4](#)

(Usually optional) Set an explicit application package name that limits the components this Intent will resolve to. If left to the default value of null, all components in all applications will be considered. If non-null, the Intent can only match the components in the given application package.

#### Parameters

*packageName*     The name of the application package to handle the intent, or null to allow any application package.

#### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

#### See Also

[getPackage\(\)](#)

[resolveActivity\(PackageManager\)](#)

public void **setSelector** ([Intent](#) selector)

Added in [API level 15](#)

Set a selector for this Intent. This is a modification to the kinds of things the Intent will match. If the selector is set, it will be used when trying to find entities that can handle the Intent, instead of the main contents of the Intent. This allows you to build an Intent containing a generic protocol while targeting it more specifically.

An example of where this may be used is with things like [CATEGORY\\_APP\\_BROWSER](#) ([/reference/android/content/Intent.html#CATEGORY\\_APP\\_BROWSER](#)). This category allows you to build an Intent that will launch the Browser application. However, the correct main entry point of an application is actually [ACTION\\_MAIN](#) ([/reference/android/content/Intent.html#ACTION\\_MAIN](#)). [CATEGORY\\_LAUNCHER](#) ([/reference/android/content/Intent.html#CATEGORY\\_LAUNCHER](#)) with [setComponent\(ComponentName\)](#) ([/reference/android/content/Intent.html#setComponent\(android.content.ComponentName\)](#)) used to specify the actual Activity to launch. If you launch the browser with something different, undesired behavior may happen if the user has previously or later launches it the normal way, since they do not match. Instead, you can build an Intent with the MAIN action (but no ComponentName yet specified) and set a selector with [ACTION\\_MAIN](#) ([/reference/android/content/Intent.html#ACTION\\_MAIN](#)) and [CATEGORY\\_APP\\_BROWSER](#) ([/reference/android/content/Intent.html#CATEGORY\\_APP\\_BROWSER](#)) to point it specifically to the browser activity.

Setting a selector does not impact the behavior of [filterEquals\(Intent\)](#) ([/reference/android/content/Intent.html#filterEquals\(android.content.Intent\)](#)) and [filterHashCode\(\)](#) ([/reference/android/content/Intent.html#filterHashCode\(\)](#)). This is part of the desired behavior of a selector -- it does not impact the base meaning of the Intent, just what kinds of things will be matched against it when determining who can handle it.

You can not use both a selector and [setPackage\(String\)](#) ([/reference/android/content/Intent.html#setPackage\(java.lang.String\)](#)) on the same base Intent.

#### Parameters



*selector*    The desired selector Intent; set to null to not use a special selector.

public void **setSourceBounds** (Rect r)

Added in [API level 7](#)

Set the bounds of the sender of this intent, in screen coordinates. This can be used as a hint to the receiver for animations and the like. Null means that there is no source bounds.

public Intent **setType** (String type)

Added in [API level 1](#)

Set an explicit MIME data type.

This is used to create intents that only specify a type and not data, for example to indicate the type of data to return.

This method automatically clears any data that was previously set (for example by [setData\(Uri\)](#) ([/reference/android/content/Intent.html#setData\(android.net.Uri\)](#))).

*Note: MIME type matching in the Android framework is case-sensitive, unlike formal RFC MIME types. As a result, you should always write your MIME types with lower case letters, or use [normalizeMimeType\(String\)](#) ([/reference/android/content/Intent.html#normalizeMimeType\(java.lang.String\)](#)) or [setTypeAndNormalize\(String\)](#) ([/reference/android/content/Intent.html#setTypeAndNormalize\(java.lang.String\)](#)) to ensure that it is converted to lower case.*

#### Parameters

*type*    The MIME type of the data being handled by this intent.

#### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

#### See Also

[getType\(\)](#)  
[setTypeAndNormalize\(String\)](#)  
[setDataAndType\(Uri, String\)](#)  
[normalizeMimeType\(String\)](#)

public Intent **setTypeAndNormalize** (String type)

Added in [API level 16](#)

Normalize and set an explicit MIME data type.

This is used to create intents that only specify a type and not data, for example to indicate the type of data to return.

This method automatically clears any data that was previously set (for example by [setData\(Uri\)](#) ([/reference/android/content/Intent.html#setData\(android.net.Uri\)](#))).

The MIME type is normalized using [normalizeMimeType\(String\)](#) ([/reference/android/content/Intent.html#normalizeMimeType\(java.lang.String\)](#)) before it is set, so really this is just a convenience method for

```
setType(Intent.normalizeMimeType(type))
```

#### Parameters

*type*    The MIME type of the data being handled by this intent.

#### Returns

Returns the same Intent object, for chaining multiple calls into a single statement.

#### See Also

[getType\(\)](#)

[setData\(Uri\)](#)  
[normalizeMimeType\(String\)](#)

public **String** **toString** ()

Added in [API level 1](#)

Returns a string containing a concise, human-readable description of this object. Subclasses are encouraged to override this method and provide an implementation that takes into account the object's type and data. The default implementation is equivalent to the following expression:

```
getClass().getName() + '@' + Integer.toHexString(hashCode())
```

See [Writing a useful toString method](#) ([/reference/java/lang/Object.html#writing\\_toString](#)) if you intend implementing your own toString method.

#### Returns

a printable representation of this object.

public **String** **toURI** ()

Added in [API level 1](#)

This method was deprecated in API level 4.

Use [toUri\(int\)](#) ([/reference/android/content/Intent.html#toUri\(int\)](#)) instead.

Call [toUri\(int\)](#) ([/reference/android/content/Intent.html#toUri\(int\)](#)) with 0 flags.

public **String** **toUri** (int flags)

Added in [API level 4](#)

Convert this Intent into a String holding a URI representation of it. The returned URI string has been properly URI encoded, so it can be used with [Uri.parse\(String\)](#) ([/reference/android/net/Uri.html#parse\(java.lang.String\)](#)). The URI contains the Intent's data as the base URI, with an additional fragment describing the action, categories, type, flags, package, component, and extras.

You can convert the returned string back to an Intent with [getIntent\(String\)](#) ([/reference/android/content/Intent.html#getIntent\(java.lang.String\)](#)).

#### Parameters

*flags* Additional operating flags. Either 0 or [URI\\_INTENT\\_SCHEME](#).

#### Returns

Returns a URI encoding URI string describing the entire contents of the Intent.

public void **writeToParcel** ([Parcel](#) out, int flags)

Added in [API level 1](#)

Flatten this object in to a Parcel.

#### Parameters

*out* The Parcel in which the object should be written.

*flags* Additional flags about how the object should be written. May be 0 or [PARCELABLE\\_WRITE\\_RETURN\\_VALUE](#).