public class Intent
   extends Object
   implements Parcelable

java.lang.Object
   \ android.content.Intent

Class Overview

An intent is an abstract description of an operation to be performed. It can be used with startActivity to launch an Activity, broadcastIntent to send it to any interested BroadcastReceiver components, and startService(Intent) or bindService(Intent, ServiceConnection, int) to communicate with a background Service.

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. The primary pieces of information in an intent are:

- **action** -- The general action to be performed, such as ACTION_VIEW, ACTION_EDIT, ACTION_MAIN, etc.
- **data** -- The data to operate on, such as a person record in the contacts database, expressed as a Uri.

Some examples of action/data pairs are:

- **ACTION_VIEW** content://contacts/1 -- Display information about the person whose identifier is "1".
- **ACTION_DIAL** content://contacts/1 -- Display the phone dialer with the person filled in.
- **ACTION_VIEW** tel:123 -- Display the phone dialer with the given number filled in. Note how the VIEW action does what is considered the most reasonable thing for a particular URI.
- **ACTION_DIAL** tel:123 -- Display the phone dialer with the given number filled in.
- **ACTION_EDIT** content://contacts/1 -- Edit information about the person whose identifier is "1".
- **ACTION_VIEW** content://contacts/ -- 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 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 means it should appear in the Launcher as a top-level application, while 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 of any additional information. This can be used to provide extended information to the component. For example, if we have a 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** with category **CATEGORY_HOME** -- Launch the home screen.
- **ACTION_GET_CONTENT** with MIME type `vnd.android.cursor.item/phone` -- 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** with MIME type `/*/*` and category **CATEGORY_OPENABLE** -- Display all pickers for data that can be opened with `ContentResolver.openInputStream()`, 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** is called "android.app.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)` or `setClass(Context, 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**, **BroadcastReceiver**, or **Service** (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** objects explicitly registered with `registerReceiver(BroadcastReceiver, IntentFilter)`.) More details on this can be found in the documentation on the **IntentFilter** 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** 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` and `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` so that they can be found by `Context.startActivity()`.

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.

```xml
<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_notes_list">
        <intent-filter>
            <action android:value="android.intent.action.MAIN" />
            <category android:value="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity class=".NoteEditor" android:label="@string/title_note">
        <intent-filter>
            <action android:value="android.intent.action.VIEW" />
            <action android:value="android.intent.action.EDIT" />
            <category android:value="android.intent.category.DEFAULT" />
            <type android:value="vnd.android.cursor.item/vnd.google.note" />
        </intent-filter>
    </activity>
    <activity class=".TitleEditor" android:label="@string/title_edit_title"
        android:theme="@android:style/Theme.Dialog">
        <intent-filter android:label="@string/resolve_title">
            <action android:value="com.android.notepad.action.EDIT_TITLE" />
            <category android:value="android.intent.category.DEFAULT" />
            <category android:value="android.intent.category.ALT"
            android:`
        </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>
2.   <action android:value="android.intent.action.MAIN" />
3.   <category android:value="android.intent.category.LAUNCHER" />
4.  </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.

5.  <intent-filter>
6.   <action android:value="android.intent.action.VIEW" />
7.   <action android:value="android.intent.action.EDIT" />
8.   <action android:value="android.intent.action.PICK" />
9.   <category android:value="android.intent.category.DEFAULT" />
10.  <type android:value="vnd.android.cursor.dir/vnd.google.note" />
11.  </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 method to resolve your activity when its component name is not explicitly specified.

12.  <intent-filter>
13.   <action android:value="android.intent.action.GET_CONTENT" />
14.   <type android:value="vnd.android.cursor.item/vnd.google.note" />
15.  </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.app.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">
2.    <action android:value="android.intent.action.VIEW" />
3.    <action android:value="android.intent.action.EDIT" />
4.    <category android:value="android.intent.category.DEFAULT" />
5.    <type android:value="vnd.android.cursor.item/vnd.google.note" />
6. </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.

7. `<intent-filter>
8.    <action android:value="android.intent.action.INSERT" />
9.    <category android:value="android.intent.category.DEFAULT" />
10.   <type android:value="vnd.android.cursor.dir/vnd.google.note" />
11.  </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.app.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:value="com.android.notepad.action.EDIT_TITLE" />
    <category android:value="android.intent.category.DEFAULT" />
    <category android:value="android.intent.category.ALTERNATIVE" />
    <category android:value="android.intent.category.SELECTED_ALTERNATIVE" />
    <type android:value="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)` method, or more often to build dynamic menu items with `addIntentOptions(int, int, int, ComponentName, Intent[], Intent, int, 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)`). The most important, and by far most frequently used, are `ACTION_MAIN` and `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)` 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

Standard Categories

These are the current standard categories that can be used to further clarify an Intent via addCategory(String).

• CATEGORY_DEFAULT
• CATEGORY_BROWSABLE
• CATEGORY_TAB
• CATEGORY_ALTERNATIVE
• CATEGORY_SELECTED_ALTERNATIVE
• CATEGORY_LAUNCHER
• CATEGORY_INFO
• CATEGORY_HOME
• CATEGORY_PREFERENCE
• CATEGORY_TEST

Standard Extra Data

These are the current standard fields that can be used as extra data via putExtra(String, Bundle).

• EXTRA_TEMPLATE
• EXTRA_INTENT
• EXTRA_STREAM
• EXTRA_TEXT

Flags

These are the possible flags that can be used in the Intent via setFlags(int) and addFlags(int). See setFlags(int) for a list of all possible flags.

Summary

Nested Classes

public class Intent.FilterComparison
  Wrapper class holding an Intent and implementing comparisons on it for the purpose of filtering.

public 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 or out of Airplane Mode.
  Activity Action: List all available applications

String ACTION_ALL_APPS
  Activity Action: List all available applications

String ACTION_ANSWER
  Activity Action: Handle an incoming phone call.

String ACTION_ATTACH_DATA
  Used to indicate that some piece of data should be attached to some
<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_BATTERY_CHANGED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Broadcast Action: The charging state, or charge level of the battery has changed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_BATTERY_LOW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Broadcast Action: Indicates low battery condition on the device.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_BOOT_COMPLETED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Broadcast Action: This is broadcast once, after the system has finished booting.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_BUG_REPORT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Activity Action: Show activity for reporting a bug.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_CALL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Activity Action: Perform a call to someone specified by the data.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_CALL_BUTTON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Activity Action: The user pressed the &quot;call&quot; button to go to the dialer or other appropriate UI for placing a call.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_CAMERA_BUTTON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Broadcast Action: The &quot;Camera Button&quot; was pressed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_CHOOSER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Activity Action: Display an activity chooser, allowing the user to pick what they want to before proceeding.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_CLOSE_SYSTEM_DIALOGS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Broadcast Action: This is broadcast when a user action should request a temporary system dialog to dismiss.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_CONFIGURATION_CHANGED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Broadcast Action: The current device Configuration (orientation, locale, etc) has changed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_CREATE_SHORTCUT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Activity Action: Creates a shortcut.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_DATE_CHANGED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Broadcast Action: The date has changed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A synonym for ACTION_VIEW, the &quot;standard&quot; action that is performed on a piece of data.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_DELETE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Activity Action: Delete the given data from its container.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_DEVICE_STORAGE_LOW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Broadcast Action: Indicates low memory condition on the device</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_DEVICE_STORAGE_OK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Broadcast Action: Indicates low memory condition on the device no longer exists</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_DIAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Activity Action: Dial a number as specified by the data.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_EDIT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Activity Action: Provide explicit editable access to the given data.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_FACTORY_TEST</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Activity Action: Main entry point for factory tests.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_GET_CONTENT</th>
</tr>
</thead>
</table>
|        | Activity Action: Allow the user to select a particular kind of data and
<table>
<thead>
<tr>
<th>String</th>
<th>ACTION_GTALK_SERVICE_CONNECTED</th>
<th>Broadcast Action: An GTalk connection has been established.</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>ACTION_GTALK_SERVICE_DISCONNECTED</td>
<td>Broadcast Action: An GTalk connection has been disconnected.</td>
</tr>
<tr>
<td>String</td>
<td>ACTION_HEADSET_PLUG</td>
<td>Broadcast Action: Wired Headset plugged in or unplugged.</td>
</tr>
<tr>
<td>String</td>
<td>ACTION_INPUT_METHOD_CHANGED</td>
<td>Broadcast Action: An input method has been changed.</td>
</tr>
<tr>
<td>String</td>
<td>ACTION_INSERT</td>
<td>Activity Action: Insert an empty item into the given container.</td>
</tr>
<tr>
<td>String</td>
<td>ACTION_INSERT_OR_EDIT</td>
<td>Activity Action: Pick an existing item, or insert a new item, and then edit it.</td>
</tr>
<tr>
<td>String</td>
<td>ACTION_MAIN</td>
<td>Activity Action: Start as a main entry point, does not expect to receive data.</td>
</tr>
<tr>
<td>String</td>
<td>ACTION_MANAGE_PACKAGE_STORAGE</td>
<td>Broadcast Action: Indicates low memory condition notification acknowledged by user and package management should be started.</td>
</tr>
<tr>
<td>String</td>
<td>ACTION_MEDIA_BAD_REMOVAL</td>
<td>Broadcast Action: External media was removed from SD card slot, but mount point was not unmounted.</td>
</tr>
<tr>
<td>String</td>
<td>ACTION_MEDIA_BUTTON</td>
<td>Broadcast Action: The &quot;Media Button&quot; was pressed.</td>
</tr>
<tr>
<td>String</td>
<td>ACTION_MEDIA_CHECKING</td>
<td>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.</td>
</tr>
<tr>
<td>String</td>
<td>ACTION_MEDIA_EJECT</td>
<td>Broadcast Action: User has expressed the desire to remove the external storage media.</td>
</tr>
<tr>
<td>String</td>
<td>ACTION_MEDIA_MOUNTED</td>
<td>Broadcast Action: External media is present and mounted at its mount point.</td>
</tr>
<tr>
<td>String</td>
<td>ACTION_MEDIA_NOFS</td>
<td>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.</td>
</tr>
<tr>
<td>String</td>
<td>ACTION_MEDIA_REMOVED</td>
<td>Broadcast Action: External media has been removed.</td>
</tr>
<tr>
<td>String</td>
<td>ACTION_MEDIA_SCANNER_FINISHED</td>
<td>Broadcast Action: The media scanner has finished scanning a directory.</td>
</tr>
<tr>
<td>String</td>
<td>ACTION_MEDIA_SCANNER_SCAN_FILE</td>
<td>Broadcast Action: Request the media scanner to scan a file and add it to the media database.</td>
</tr>
<tr>
<td>String</td>
<td>Broadcast Action</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>ACTION_MEDIA_SCANNER_STARTED</td>
<td>The media scanner has started scanning a directory.</td>
<td></td>
</tr>
<tr>
<td>ACTION_MEDIA_SHARED</td>
<td>External media is unmounted because it is being shared via USB mass storage.</td>
<td></td>
</tr>
<tr>
<td>ACTION_MEDIA_UNMOUNTABLE</td>
<td>External media is present but cannot be mounted.</td>
<td></td>
</tr>
<tr>
<td>ACTION_MEDIA_UNMOUNTED</td>
<td>External media is present, but not mounted at its mount point.</td>
<td></td>
</tr>
<tr>
<td>ACTION_NEW_OUTGOING_CALL</td>
<td>An outgoing call is about to be placed.</td>
<td></td>
</tr>
<tr>
<td>ACTION_PACKAGE_ADDED</td>
<td>A new application package has been installed on the device.</td>
<td></td>
</tr>
<tr>
<td>ACTION_PACKAGE_CHANGED</td>
<td>An existing application package has been changed (e.g.</td>
<td></td>
</tr>
<tr>
<td>ACTION_PACKAGE_DATA_CLEARED</td>
<td>The user has cleared the data of a package.</td>
<td></td>
</tr>
<tr>
<td>ACTION_PACKAGE_INSTALL</td>
<td>Trigger the download and eventual installation of a package.</td>
<td></td>
</tr>
<tr>
<td>ACTION_PACKAGE_REMOVED</td>
<td>An existing application package has been removed from the device.</td>
<td></td>
</tr>
<tr>
<td>ACTION_PACKAGE_REPLACED</td>
<td>A new version of an application package has been installed, replacing an existing version that was previously installed.</td>
<td></td>
</tr>
<tr>
<td>ACTION_PACKAGE_RESTARTED</td>
<td>The user has restarted a package, and all of its processes have been killed.</td>
<td></td>
</tr>
<tr>
<td>ACTION_PICK</td>
<td>Activity: Pick an item from the data, returning what was selected.</td>
<td></td>
</tr>
<tr>
<td>ACTION_PICK_ACTIVITY</td>
<td>Activity: Pick an activity given an intent, returning the class selected.</td>
<td></td>
</tr>
<tr>
<td>ACTION_PROVIDER_CHANGED</td>
<td>Some content providers have parts of their namespace where they publish new events or items that the user may be especially interested in.</td>
<td></td>
</tr>
<tr>
<td>ACTION_REBOOT</td>
<td>Have the device reboot.</td>
<td></td>
</tr>
<tr>
<td>ACTION_RUN</td>
<td>Run the data, whatever that means.</td>
<td></td>
</tr>
<tr>
<td>ACTION_SCREEN_OFF</td>
<td>Sent after the</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>ACTION_SCREEN_ON</td>
<td>Broadcast Action: Sent after the screen turns on.</td>
<td></td>
</tr>
<tr>
<td>ACTION_SEARCH</td>
<td>Activity Action: Perform a search.</td>
<td></td>
</tr>
<tr>
<td>ACTION_SEARCH_LONG_PRESS</td>
<td>Activity Action: Start action associated with long pressing on the search key.</td>
<td></td>
</tr>
<tr>
<td>ACTION_SEND</td>
<td>Activity Action: Deliver some data to someone else.</td>
<td></td>
</tr>
<tr>
<td>ACTION_SENDTO</td>
<td>Activity Action: Send a message to someone specified by the data.</td>
<td></td>
</tr>
<tr>
<td>ACTION_SET_WALLPAPER</td>
<td>Activity Action: Show settings for choosing wallpaper</td>
<td></td>
</tr>
<tr>
<td>ACTION_SYNC</td>
<td>Activity Action: Perform a data synchronization.</td>
<td></td>
</tr>
<tr>
<td>ACTION_SYSTEM_TUTORIAL</td>
<td>Activity Action: Start the platform-defined tutorial</td>
<td></td>
</tr>
<tr>
<td>ACTION_TIMEZONE_CHANGED</td>
<td>Broadcast Action: The timezone has changed.</td>
<td></td>
</tr>
<tr>
<td>ACTION_TIME_CHANGED</td>
<td>Broadcast Action: The time was set.</td>
<td></td>
</tr>
<tr>
<td>ACTION_TIME_TICK</td>
<td>Broadcast Action: The current time has changed.</td>
<td></td>
</tr>
<tr>
<td>ACTION_UID_REMOVED</td>
<td>Broadcast Action: A user ID has been removed from the system.</td>
<td></td>
</tr>
<tr>
<td>ACTION_UMS_CONNECTED</td>
<td>Broadcast Action: The device has entered USB Mass Storage mode.</td>
<td></td>
</tr>
<tr>
<td>ACTION_UMS_DISCONNECTED</td>
<td>Broadcast Action: The device has exited USB Mass Storage mode.</td>
<td></td>
</tr>
<tr>
<td>ACTION_USER_PRESENT</td>
<td>Broadcast Action: Sent when the user is present after device wakes up (e.g. when the keyguard is gone).</td>
<td></td>
</tr>
<tr>
<td>ACTION_VIEW</td>
<td>Activity Action: Display the data to the user.</td>
<td></td>
</tr>
<tr>
<td>ACTION_VOICE_COMMAND</td>
<td>Activity Action: Start Voice Command.</td>
<td></td>
</tr>
<tr>
<td>ACTION_WALLPAPER_CHANGED</td>
<td>Broadcast Action: The current system wallpaper has changed.</td>
<td></td>
</tr>
<tr>
<td>ACTION_WEB_SEARCH</td>
<td>Activity Action: Perform a web search.</td>
<td></td>
</tr>
<tr>
<td>CATEGORY_ALTERNATIVE</td>
<td>Set if the activity should be considered as an alternative action to the data the user is currently viewing.</td>
<td></td>
</tr>
<tr>
<td>CATEGORY_BROWSABLE</td>
<td>Activities that can be safely invoked</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>CATEGORY_DEFAULT</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>CATEGORY_DEVELOPMENT_PREFERENCE</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>CATEGORY_EMBED</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>CATEGORY_FRAMEWORK_INSTRUMENTATION_TEST</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>CATEGORY_HOME</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>CATEGORY_INFO</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>CATEGORY_LAUNCHER</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>CATEGORY_MONKEY</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>CATEGORY_OPENABLE</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>CATEGORY_PREFERENCE</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>CATEGORY_SAMPLE_CODE</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>CATEGORY_SELECTED_ALTERNATIVE</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>CATEGORY_TAB</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>CATEGORY_TEST</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>CATEGORY_UNIT_TEST</td>
<td></td>
</tr>
<tr>
<td>Creator&lt;Intent&gt;</td>
<td>CREATOR</td>
<td></td>
</tr>
<tr>
<td>String</td>
<td>EXTRA_ALARM_COUNT</td>
<td></td>
</tr>
</tbody>
</table>

from a browser must support this category.
Set if the activity should be an option for the default action (center press) to perform on a piece of data.
This activity is a development preference panel.
Capable of running inside a parent activity container.
To be used as code under test for framework instrumentation tests.
This is the home activity, that is the first activity that is displayed when the device boots.
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 in the all apps list.
Should be displayed in the top-level launcher.
This activity may be exercised by the monkey or other automated test tools.
Used to indicate that a GET_CONTENT intent only wants URIs that can be opened with ContentResolver.openInputStream.
This activity is a preference panel.
To be used as a sample code example (not part of the normal user experience).
Set if the activity should be considered as an alternative selection action to the data the user has currently selected.
Intended to be used as a tab inside of an containing TabActivity.
To be used as a test (not part of the normal user experience).
To be used as a unit test (run through the Test Harness).
Used as an int extra field in AlarmManager intents to tell the application being invoked how many pending alarms are being delivered with the intent.
<table>
<thead>
<tr>
<th>String</th>
<th>EXTRA_BCC</th>
<th>A String[] holding e-mail addresses that should be blind carbon copied.</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>EXTRA_CC</td>
<td>A String[] holding e-mail addresses that should be carbon copied.</td>
</tr>
<tr>
<td></td>
<td>EXTRA_DATA_REMOVED</td>
<td>Used as a boolean extra field in <code>ACTION_PACKAGE_REMOVED</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).</td>
</tr>
<tr>
<td></td>
<td>EXTRA_DONT_KILL_APP</td>
<td>Used as a boolean extra field in <code>ACTION_PACKAGE_REMOVED</code> or <code>ACTION_PACKAGE_CHANGED</code> intents to override the default action of restarting the application.</td>
</tr>
<tr>
<td>String</td>
<td>EXTRA_EMAIL</td>
<td>A String[] holding e-mail addresses that should be delivered to.</td>
</tr>
<tr>
<td>String</td>
<td>EXTRA_INTENT</td>
<td>An Intent describing the choices you would like shown with <code>ACTION_PICK_ACTIVITY</code>.</td>
</tr>
<tr>
<td>String</td>
<td>EXTRA_KEY_EVENT</td>
<td>A KeyEvent object containing the event that triggered the creation of the Intent it is in.</td>
</tr>
<tr>
<td>String</td>
<td>EXTRA_PHONE_NUMBER</td>
<td>A String holding the phone number originally entered in <code>ACTION_NEW_OUTGOING_CALL</code>, or the actual number to call in an <code>ACTION_CALL</code>.</td>
</tr>
<tr>
<td></td>
<td>EXTRA_REPLACING</td>
<td>Used as a boolean extra field in <code>ACTION_PACKAGE_REMOVED</code> 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.</td>
</tr>
<tr>
<td>String</td>
<td>EXTRA_SHORTCUT_ICON</td>
<td>The name of the extra used to define the icon, as a Bitmap, of a shortcut.</td>
</tr>
<tr>
<td>String</td>
<td>EXTRA_SHORTCUT_ICONRESOURCE</td>
<td>The name of the extra used to define the icon, as a ShortcutIconResource, of a shortcut.</td>
</tr>
<tr>
<td>String</td>
<td>EXTRA_SHORTCUT_INTENT</td>
<td>The name of the extra used to define the Intent of a shortcut.</td>
</tr>
<tr>
<td>String</td>
<td>EXTRA_SHORTCUT_NAME</td>
<td>The name of the extra used to define the name of a shortcut.</td>
</tr>
<tr>
<td>String</td>
<td>EXTRA_STREAM</td>
<td>A content: URI holding a stream of data associated with the Intent, used with <code>ACTION_SEND</code> to supply the data being sent.</td>
</tr>
<tr>
<td>String</td>
<td>EXTRA_SUBJECT</td>
<td>A constant string holding the desired</td>
</tr>
</tbody>
</table>
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 intents 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_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 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_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) old activity as a new Intent.

int FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET

If set, this marks a point in the task's activity stack that should be cleared when the task is reset.

int FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS

If set, the new activity is not kept in the list of recently launched activities.

int FLAG_ACTIVITY_FORWARD_RESULT

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.

int FLAG_ACTIVITY_LAUNCHED_FROM_HISTORY

This flag is not normally set by
<table>
<thead>
<tr>
<th>int</th>
<th>Constant Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>FLAG_ACTIVITY_MULTIPLE_TASK</td>
<td>application code, but set for you by the system if this activity is being launched from history (longpress home key).&lt;br&gt;&lt;br&gt;<strong>Do not use this flag unless you are implementing your own top-level application launcher.</strong>&lt;br&gt;&lt;br&gt;If set, this activity will become the start of a new task on this history stack.</td>
</tr>
<tr>
<td>int</td>
<td>FLAG_ACTIVITY_NEW_TASK</td>
<td>If set, the new activity is not kept in the history stack.</td>
</tr>
<tr>
<td>int</td>
<td>FLAG_ACTIVITY_NO_HISTORY</td>
<td>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.</td>
</tr>
<tr>
<td>int</td>
<td>FLAG_ACTIVITY_NO_USER_ACTION</td>
<td>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.</td>
</tr>
<tr>
<td>int</td>
<td>FLAG_ACTIVITY_PREVIOUS_IS_TOP</td>
<td>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.</td>
</tr>
<tr>
<td>int</td>
<td>FLAG_ACTIVITY_RESET_TASK_IF_NEEDED</td>
<td>If set, the activity will not be launched if it is already running at the top of the history stack. 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. Can be set by the caller to indicate that this Intent is coming from a background operation, not from direct user interaction.</td>
</tr>
<tr>
<td>int</td>
<td>FLAG_ACTIVITY_SINGLE_TOP</td>
<td>If set, the recipient of this Intent will be granted permission to perform read operations on the Uri in the Intent's data.</td>
</tr>
<tr>
<td>int</td>
<td>FLAG_DEBUG_LOG_RESOLUTION</td>
<td>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.</td>
</tr>
<tr>
<td>int</td>
<td>FLAG_FROM_BACKGROUND</td>
<td>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.</td>
</tr>
<tr>
<td>int</td>
<td>FLAG_GRANT_READ_URI_PERMISSION</td>
<td>If set, the activity will not be launched if it is already running at the top of the history stack. 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. Can be set by the caller to indicate that this Intent is coming from a background operation, not from direct user interaction.</td>
</tr>
<tr>
<td>int</td>
<td>FLAG_GRANT_READ_URI_PERMISSION</td>
<td>If set, the recipient of this Intent will be granted permission to perform read operations on the Uri in the Intent's data.</td>
</tr>
</tbody>
</table>
int FLAG_GRANT_WRITE_URI_PERMISSION
If set, the recipient of this Intent will be granted permission to perform write operations on the Uri in the Intent's data.

int FLAG_RECEIVER_REGISTERED_ONLY
If set, when sending a broadcast only registered receivers will be called -- no BroadcastReceiver components will be launched.

Inherited Constants

Int From interface android.os.Parcelable

int CONTENTS_FILE_DESCRIPTOR
Bit masks for use with describeContents(): each bit represents a kind of object considered to have potential special significance when marshalled.

Flag for use with writeToParcel(Parcel, int): the object being written is a return value, that is the result of a function such as "Parcelable someFunction()", "void someFunction(out Parcelable)", or "void someFunction(inout Parcelable)".

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() 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)
Determine if two intents are the same for the purposes of intent resolution (filtering).

```java
int filterHashCode()
Generate hash code that matches semantics of filterEqual().

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
getCharSequenceExtra(String name)
Retrieve extended data from the 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.

static Intent
getIntent(String uri)
Retrieve extended data from the intent.
```
Create an intent from a URI.

static Intent getIntentOld(String uri)

Retrieve extended data from the intent.

long[] getLongArrayExtra(String name)

long getLongExtra(String name, long defaultValue)

Parcelable[] getParcelableArrayExtra(String name)

<T extends Parcelable> ArrayList<T> getParcelableArrayListExtra(String name)

<T extends Parcelable> T getParcelableExtra(String name)

String getScheme()

Retrieve extended data from the intent.

Serializable getSerializableExtra(String name)

short[] getShortArrayExtra(String name)

short getShortExtra(String name, short defaultValue)

String[] getStringArrayExtra(String name)

ArrayList<String> getStringArrayListExtra(String name)

String getStringExtra(String name)

Retrieve any explicit MIME type included in the intent.

String getType()

Check if an category exists in the intent.

boolean hasCategory(String category)

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 parseIntent(Resources resources, XmlPullParser parser, AttributeSet attrs)

Parses the "intent" element (and its children) from XML and instantiates an Intent object.

Intent putExtra(String name, String[] value)

Add extended data to the intent.

Intent putExtra(String name, Parcelable value)

Add extended data to the intent.

Intent putExtra(String name, long value)

Add extended data to the intent.

Intent putExtra(String name, boolean value)

Add extended data to the intent.

Intent putExtra(String name, double value)

Add extended data to the intent.

Intent putExtra(String name, Parcelable[] value)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Intent
putExtras(Intent src)
Copy all extras in 'src' in to this intent.

Intent
putExtras(Bundle extras)
Add a set of extended data to the intent.

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

Intent
putParcelableArrayListExtra(String name, ArrayList<? extends Parcelable> value)
Add extended data to the intent.

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

void
readFromParcel(Parcel in)

void
removeCategory(String category)
Remove an category from an intent.

void
removeExtra(String name)
Remove extended data from the intent.
replaceExtras(Intent src)
Completely replace the extras in the Intent with the extras in the given Intent.

replaceExtras(Bundle extras)
Completely replace the extras in the Intent with the given Bundle of extras.

resolveActivity(PackageManager pm)
Return the Activity component that should be used to handle this intent.

resolveActivityInfo(PackageManager pm, int flags)
Resolve the Intent into an ActivityInfo describing the activity that should execute the intent.

resolveType(ContentResolver resolver)
Return the MIME data type of this intent.

resolveType(Context context)
Return the MIME data type of this intent.

resolveTypeIfNeeded(ContentResolver resolver)
Return the MIME data type of this intent, only if it will be needed for intent resolution.

setAction(String action)
Set the general action to be performed.

setComponent(ComponentName component)
(Usually optional) Explicitly set the component to handle the intent.

setData(Uri data)
Set the data this intent is operating on.

setDataAndType(Uri data, String type)
(Usually optional) Set the data for the intent along with an explicit MIME data type.

setExtrasClassLoader(ClassLoader loader)
Sets the ClassLoader that will be used when unmarshalling any Parcelable values from the extras of this Intent.

setFlags(int flags)
Set special flags controlling how this intent is handled.

setType(String type)
Set an explicit MIME data type.

toString()
Returns a string containing a concise, human-readable description of this object.

toURI()
Flatten this object in to a Parcel.

Object
clone()
Creates and returns a copy of this Object.
boolean equals(Object o)

Compares this instance with the specified object and indicates if they are equal.

void finalize()

Is called before the object's memory is being reclaimed by the VM.

final Class<? extends Object> getClass()

Returns the unique instance of Class which represents this object's class.

int hashCode()

Returns an integer hash code for this object.

final void notify()

Causes a thread which is waiting on this object's monitor (by means of calling one of the
wait() methods) to be woken up.

final void notifyAll()

Causes all threads which are waiting on this object's monitor (by means of calling one of
the wait() methods) to be woken up.

String toString()

Returns a string containing a concise, human-readable description of this object.

final void wait(long millis, int nanos)

Causes the calling thread to wait until another thread calls the notify() or notifyAll()
method of this object or until the specified timeout expires.

final void wait(long millis)

Causes the calling thread to wait until another thread calls the notify() or notifyAll()
method of this object or until the specified timeout expires.

final void wait()

Causes the calling thread to wait until another thread calls the notify() or notifyAll()
method of this object.

From interface android.os.Parcelable

abstract int describeContents()

Describe the kinds of special objects contained in this Parcelable's marshalled representation.

abstract void writeToParcel(Parcel dest, int flags)

Flatten this object in to a Parcel.

### Constants

**public static final String ACTION_AIRPLANE_MODE_CHANGED**

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

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

**public static final String ACTION_ALL_APPS**

Activity Action: List all available applications

Input: Nothing.
Output: nothing.

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

```
public static final String ACTION_ANSWER

Activity Action: Handle an incoming phone call.

Input: nothing.

Output: nothing.
```

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

```
public static final String ACTION_ATTACH_DATA

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() is URI of data to be attached.

Output: nothing.
```

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

```
public static final String ACTION_BATTERY_CHANGED

Broadcast Action: The charging state, or charge level of the battery has changed.

You can not receive this through components declared in manifests, only by explicitly registering for it with Context.registerReceiver().

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

```
public static final String ACTION_BATTERY_LOW

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

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

```
public static final String ACTION_BOOT_COMPLETED

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 permission in order to receive this broadcast.

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

```
public static final String ACTION_BUG_REPORT

```

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

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

Input: If nothing, an empty dialer is started; else 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.

Note: this Intent cannot be used to call emergency numbers. Applications can dial emergency numbers using ACTION_DIAL, however.

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

public static final String ACTION_CALL_BUTTON

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

Broadcast Action: The "Camera Button" was pressed. Includes a single extra field, EXTRA_KEY_EVENT, containing the key event that caused the broadcast.

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

public static final String ACTION_CHOOSER

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:"

As a convenience, an Intent of this form can be created with the `createChooser(Intent, CharSequence)` function.

Input: No data should be specified. get*Extra must have a EXTRA_INTENT field containing the Intent being executed, and can optionally have a EXTRA_TITLE field containing the title text to display in the chooser.

Output: Depends on the protocol of EXTRA_INTENT.

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

```
public static final String ACTION_CLOSE_SYSTEM_DIALOGS
```

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
```

Broadcast Action: The current device Configuration (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.

See Also

- Configuration

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

```
public static final String ACTION_CREATE_SHORTCUT
```

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

Broadcast Action: The date has changed.

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

public static final String ACTION_DEFAULT

A synonym for 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

Activity Action: Delete the given data from its container.

Input: getData() is URI of data to be deleted.

Output: nothing.

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

public static final String ACTION_DEVICE_STORAGE_LOW

Broadcast Action: Indicates low memory condition on the device

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

public static final String ACTION_DEVICE_STORAGE_OK

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

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

public static final String ACTION_DIAL

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() 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_EDIT**

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

Input: `getData()` is URI of data to be edited.

Output: nothing.

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

**public static final String ACTION_FACTORY_TEST**

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**

Activity Action: Allow the user to select a particular kind of data and return it. This is different than `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. A `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 browser over the web and download the desired data, etc.

There are two main ways to use this action: if you want an 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)`. 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)`), 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` in the Intent.

Input: `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` if you can only accept data that can be opened as a stream.

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

Broadcast Action: An GTalk connection has been established.

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

public static final String ACTION_GTALK_SERVICE_DISCONNECTED

Broadcast Action: An GTalk connection has been disconnected.

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

public static final String ACTION_HEADSET_PLUG

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

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

public static final String ACTION_INPUT_METHOD_CHANGED

Broadcast Action: An input method has been changed.

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

public static final String ACTION_INSERT

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

Input: 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

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

Input: 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_MAIN**

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_PACKAGE_STORAGE**

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**

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**

Broadcast Action: The "Media Button" was pressed. Includes a single extra field, 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**

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**

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**

Broadcast Action: External media is present and mounted at its mount point. The path to the mount point for the removed 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.
public static final 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.

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

public static final String ACTION_MEDIA_REMOVED

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

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

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

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

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

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

public static final String ACTION_MEDIA_UNMOUNTABLE

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

Constant Value: "android.intent.action.MEDIA_UNMOUNTABLE"
public static final String ACTION_MEDIA_UNMOUNTED

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

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

public static final String ACTION_NEW_OUTGOING_CALL

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.

You must hold the PROCESS_OUTGOING_CALLS permission to receive this Intent.

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

public static final String ACTION_PACKAGE_ADDED

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.

My 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.

Constant Value: "android.intent.action_PACKAGE_ADDED"

public static final String ACTION_PACKAGE_CHANGED
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.

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

**public static final String ACTION_PACKAGE_DATA_CLEARED**

Broadcast Action: The user has cleared the data of a package. This should be preceded by **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.

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

**public static final String ACTION_PACKAGE_INSTALL**

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

Input: **getData()** is the URI of the package file to download.

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

**public static final String ACTION_PACKAGE_REMOVED**

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.

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

**public static final String ACTION_PACKAGE_REPLACED**

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.

My include the following extras:

- **EXTRA_UID** containing the integer uid assigned to the new package.

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

**public static final String ACTION_PACKAGE_RESTARTED**
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.

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

**public static final String ACTION_PICK**

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

Input: `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**

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

Input: get*Extra field **EXTRA_INTENT** is an Intent used with `queryIntentActivities(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_PROVIDER_CHANGED**

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

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

public static final String ACTION_RUN

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.

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

public static final String ACTION_SCREEN_ON

Broadcast Action: Sent after the screen turns on.

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

public static final String ACTION_SEARCH

Activity Action: Perform a search.

Input: getStringExtra(SearchManager.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

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

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)`), 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()` is the MIME type of the data being sent. `getExtra` can have either an `EXTRA_TEXT` or `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).

Optional standard extras, which may be interpreted by some recipients as appropriate, are: `EXTRA_EMAIL`, `EXTRA_CC`, `EXTRA_BCC`, `EXTRA_SUBJECT`.

Output: nothing.

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

`public static final String ACTION_SENDTO`  
Activity Action: Send a message to someone specified by the data.

Input: `getData()` is URI describing the target.

Output: nothing.

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

`public static final String ACTION_SET_WALLPAPER`  
Activity Action: Show settings for choosing wallpaper

Input: Nothing.

Output: Nothing.

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

`public static final String ACTION_SYNC`  
Activity Action: Perform a data synchronization.

Input: ?

Output: ?

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

`public static final String ACTION_SYSTEM_TUTORIAL`  
Activity Action: Start the platform-defined tutorial
Input: `getStringExtra(SearchManager.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.

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

**public static final String ACTION_TIME_CHANGED**

Broadcast Action: The time was set.

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

**public static final String ACTION_TIME_TICK**

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()`.

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

**public static final String ACTION_UID_REMOVED**

Broadcast Action: A user ID has been removed from the system. The user ID number is stored in the extra data under `EXTRA_UID`.

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

**public static final String ACTION_UMS_CONNECTED**

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**

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_USER_PRESENT

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

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

public static final String ACTION_VIEW

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() is URI from which to retrieve data.

Output: nothing.

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

public static final String ACTION_VOICE_COMMAND

Activity Action: Start Voice Command.

Input: Nothing.

Output: Nothing.

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

public static final String ACTION_WALLPAPER_CHANGED

Broadcast Action: The current system wallpaper has changed. See getWallpaper() for retrieving the new wallpaper.

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

public static final String ACTION_WEB_SEARCH

Activity Action: Perform a web search.

Input: getStringExtra(SearchManager.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

Set if the activity should be considered as an alternative action to the data the user is currently viewing. See also 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`, 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_BROWSABLE**

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_DEFAULT**

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 normal - 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_DEVELOPMENT_PREFERENCE**

This activity is a development preference panel.

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

**public static final String CATEGORY_EMBED**

Capable of running inside a parent activity container.

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

**public static final String CATEGORY_FRAMEWORK_INSTRUMENTATION_TEST**

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

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

**public static final String CATEGORY_HOME**

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

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 in the all apps list.

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

public static final String CATEGORY_LAUNCHER

Should be displayed in the top-level launcher.

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

public static final String CATEGORY_MONKEY

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

Used to indicate that a GET_CONTENT intent only wants URIs that can be opened with ContentResolver.openInputStream. Openable URIs must support the columns in OpenableColumns when queried, though it is allowable for those columns to be blank.

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

public static final String CATEGORY_PREFERENCE

This activity is a preference panel.

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

public static final String CATEGORY_SAMPLE_CODE

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

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

public static final String CATEGORY_SELECTED_ALTERNATIVE

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

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

**public static final String CATEGORY_TEST**

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**

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

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

**public static final Creator<Intent> CREATOR**

**public static final String EXTRA_ALARM_COUNT**

Used as an int extra field in AlarmManager 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_BCC**

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

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

**public static final String EXTRA_CC**

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

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

**public static final String EXTRA_DATA_REMOVED**

Used as a boolean extra field in 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_DONT_KILL_APP**

Used as an boolean extra field in ACTION_PACKAGE_REMOVED or 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

A String[] holding e-mail addresses that should be delivered to.

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

public static final String EXTRA_INTENT

An Intent describing the choices you would like shown with ACTION_PICK_ACTIVITY.

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

public static final String EXTRA_KEY_EVENT

A KeyEvent 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_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.

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

public static final String EXTRA_REPLACING

Used as a boolean extra field in 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_SHORTCUT_ICON

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

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

See Also

• ACTION_CREATE_SHORTCUT
• IntentShortcutIconResource
Constant Value: "android.intent.extra.shortcut.ICON_RESOURCE"

`public static final String EXTRA_SHORTCUT_INTENT`

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`

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_STREAM`

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

Constant Value: "android.intent.extra_STREAM"

`public static final String EXTRA_SUBJECT`

A constant string holding the desired subject line of a message.

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

`public static final String EXTRA_TEMPLATE`

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

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

`public static final String EXTRA_TEXT`

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

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

`public static final String EXTRA_TITLE`

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

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

**public static final String EXTRA_UID**

Used as an int extra field in **ACTION_UID_REMOVED** intents to supply the uid the package had been assigned. Also an optional extra in **ACTION_PACKAGE_REMOVED** or **ACTION_PACKAGE_CHANGED** for the same purpose.

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

**public static final int FILL_IN_ACTION**

Use with ```fillIn(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**

Use with ```fillIn(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_COMPONENT**

Use with ```fillIn(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**

Use with ```fillIn(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 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.

Constant Value: 4194304 (0x00400000)

**public static final 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) 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 task 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) it will be finished and re-created; for all other launch modes it will receive the Intent in the current instance.

This launch mode can also be used to good effect in conjunction with 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 Application Fundamentals: Activities and Tasks for more details on tasks.

Constant Value: 67108864 (0x04000000)

public static final int FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET

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 broad to the foreground with 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 proceeded 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

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

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) 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

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).
public static final int FLAG_ACTIVITY_MULTIPLE_TASK

Do not use this flag unless you are implementing your own top-level application launcher. Used in conjunction with 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 is not set.

See Application Fundamentals: Activities and Tasks for more details on tasks.

Constant Value: 134217728 (0x08000000)

public static final int FLAG_ACTIVITY_NEW_TASK

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 Application Fundamentals: Activities and Tasks for more details on 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 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_HISTORY

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 attribute.

Constant Value: 1073741824 (0x40000000)

public static final int FLAG_ACTIVITY_NO_USER_ACTION

If set, this flag will prevent the normal 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`, 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`

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`

If set in an Intent passed to `Context.startActivity()`, 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` is also specified.

Constant Value: 131072 (0x00020000)

`public static final int FLAG_ACTIVITY_RESET_TASK_IF_NEEDED`

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`

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_DEBUG_LOG_RESOLUTION`

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_FROM_BACKGROUND`

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_READ_URI_PERMISSION`  
If set, the recipient of this Intent will be granted permission to perform read operations on the Uri in the Intent's data.

Constant Value: 1 (0x00000001)

`public static final int FLAG_GRANT_WRITE_URI_PERMISSION`  
If set, the recipient of this Intent will be granted permission to perform write operations on the Uri in the Intent's data.

Constant Value: 2 (0x00000002)

`public static final int FLAG_RECEIVER_REGISTERED_ONLY`  
If set, when sending a broadcast only registered receivers will be called -- no BroadcastReceiver components will be launched.

Constant Value: 1073741824 (0x40000000)

**Public Constructors**

`public Intent ()`  
Create an empty intent.

`public Intent (Intent o)`  
Copy constructor.

`public Intent (String action)`  
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)`  
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)**

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) 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)**

Create an intent for a specific component with a specified action and data. This is equivalent using Intent(String, android.net.Uri) to construct the Intent and then calling setClass(Context, 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 is perform. 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)

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 ()

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 ()
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)**

Convenience function for creating a ACTION_CHOOSER Intent.

**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 ()**

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)**

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).
- component, as set by setComponent(ComponentName) or related methods.
- each top-level name in the associated extras.

In addition, you can use the FILL_IN_ACTION, FILL_IN_DATA, FILL_IN_CATEGORIES, and FILL_IN_COMPONENT to override the restriction where the corresponding field will not be replaced if it is already set.

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**, and **FILL IN COMPONENT** indicating which fields were changed.

public boolean filterEquals (Intent other)

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 ()

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 ()

Retrieve the general action to be performed, such as **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().

Returns

• The action of this intent or null if none is specified.

See Also

• setAction(String)

public boolean[] getBooleanArrayExtra (String name)

Retrieve extended data from the intent.

Parameters

name The name of the desired item.

Returns
public boolean getBooleanExtra (String name, boolean defaultValue)

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)

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)

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.
public byte getByteExtra (String name, byte defaultValue)

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 ()

Return the set of all categories in the intent. If there are no categories, returns NULL.

Returns

• Set 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[])
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** getCharSequenceExtra **(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 CharSequence value was found.

See Also

- putExtra(String, CharSequence)

public **ComponentName** getComponent **()**

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 **()**

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)`

```java
public String getDataString ()
```

The same as `getData()`, but returns the URI as an encoded String.

```java
public double[] getDoubleArrayExtra (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 double array value was found.

See Also

- `putExtra(String, double[])`

```java
public double getDoubleExtra (String name, double defaultValue)
```

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, double)`

```java
public Bundle getExtras ()
```

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 ()

Retrieve any special flags associated with this intent. You will normally just set them with 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)

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 float array value was found.

See Also

• putExtra(String, float[])

public float getFloatExtra (String name, float defaultValue)

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 no such item is present

See Also

• putExtra(String, float)

public int[] getIntArrayExtra (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 int array value was found.

**See Also**

- putExtra(String, int[])

**public int getIntExtra (String name, int defaultValue)**

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)**

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)**

Create an intent from a URI. This URI may encode the action, category, and other intent fields, if it was returned by toURI(). 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.

Returns

• Intent The newly created Intent object.

Throws

URISyntaxException

See Also

• toURI()

public static Intent getIntentOld (String uri)

Throws

URISyntaxException

public long[] getLongArrayExtra (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 long array value was found.

See Also

• putExtra(String, long[])

public long getLongExtra (String name, long defaultValue)

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 `Parcelable[]` `getParcelableArrayExtra(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 Parcelable[] value was found.

See Also

• `putExtra(String, Parcelable[])`

public `ArrayList<T>` `getParcelableArrayListExtra(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 ArrayList value was found.

See Also

• `putParcelableArrayListExtra(String, ArrayList)`

public `T` `getParcelableExtra(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 Parcelable value was found.

See Also
public **String** getScheme ()

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()
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, short)`

public `String[]` `getStringArrayExtra` (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 String array value was found.

See Also

- `putExtra(String, String[])`

public `ArrayList<String>` `getStringArrayListExtra` (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 ArrayList value was found.

See Also

- `putStringArrayListExtra(String, ArrayList)`

public `String` `getStringExtra` (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 String value was found.

**See Also**

- [putExtra(String, String)]

```java
public String getType ()
```

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)]

```java
public boolean hasCategory (String category)
```

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)]

```java
public boolean hasExtra (String name)
```

Returns true if an extra value is associated with the given name.

**Parameters**

**name** the extra's name
public boolean hasFileDescriptors()

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 parseIntent(Resources resources, XmlPullParser parser, AttributeSet attrs)

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 Intent putExtra(String name, String[] value)

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, Parcelable value)

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, long value)

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, boolean value)

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, double value)**

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, Parcelable[] value)**

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)
public Intent putExtra (String name, char value)

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

- putExtra(Intent)
- removeExtra(String)
- getCharExtra(String, char)

public Intent putExtra (String name, int[] value)

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

- putExtra(Intent)
- removeExtra(String)
- getIntArrayExtra(String)

public Intent putExtra (String name, int value)

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, double[] value)

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, float value)

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, short value)

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, long[] value)

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, boolean[] value)

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)

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)

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, Serializable value)
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, float[] value)

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, Bundle value)

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.
public Intent putExtra (String name, byte[] value)

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)
- getBundleExtra(String)

public Intent putExtra (String name, CharSequence value)

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)

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, byte value)**

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)`
- `getByteExtra(String, byte)`

**public Intent putExtras (Intent src)**

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)**

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)`

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)`

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)`
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)`

```java
public void readFromParcel (Parcel in)
```

```java
public void removeCategory (String category)
```

Remove a category from an intent.

**Parameters**

**category** The category to remove.

**See Also**

- `addCategory(String)`

```java
public void removeExtra (String name)
```

Remove extended data from the intent.

**See Also**

- `putExtra(String, Bundle)`

```java
public Intent replaceExtras (Intent src)
```

Completely replace the extras in the Intent with the extras in the given Intent.

**Parameters**

The exact extras contained in this Intent are copied into the target intent, replacing any that were previously there.

```java
public Intent replaceExtras (Bundle extras)
```
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 ComponentName resolveActivity (PackageManager pm)**

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()` returns an explicit class, that is returned without any further consideration.
- The activity must handle the `CATEGORY_DEFAULT` Intent category to be considered.
- If `getAction()` is non-NULL, the activity must handle this action.
- If `resolveType(ContentResolver)` returns non-NULL, the activity must handle this type.
- If `addCategory(String)` has added any categories, the activity must handle ALL of the categories specified.
- 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)` 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)**

Resolve the Intent into an `ActivityInfo` describing the activity that should execute the intent. Resolution follows the same rules as described for `resolveActivity(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)

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)

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)

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)

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)

Convenience for calling setComponent(ComponentName) with the name returned by a Class object.

Parameters

- packageContext A Context of the application package implementing this class.
- cls The class name to set, equivalent to setName(context, cls.getName()).

Returns

- Returns the same Intent object, for chaining multiple calls into a single statement.

See Also

- setComponent(ComponentName)

public Intent setClassName (String packageName, String className)

Convenience for calling setComponent(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 Intent setClassName (Context packageContext, String className)

Convenience for calling setComponent(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 setComponent (ComponentName component)

(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)

Set the data this intent is operating on. This method automatically clears any type that was previously set by setType(String).

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

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)
- setDataAndType(Uri, String)

public Intent setDataAndType(Uri data, String type)

(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, Uri scheme, and host name 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, and normalize any MIME types or Uris you receive from outside of Android to ensure these elements are lower case before supplying them here.

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

- setData(Uri)
- setType(String)

public void setExtrasClassLoader(ClassLoader loader)

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)**

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()` and the FLAG_RECEIVER_* flags are all for use with `Context.sendBroadcast()`.

See the Application Fundamentals: Activities and Tasks 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_WHEN_TASK_RESET`
- `FLAG_ACTIVITY_CLEAR_TOP`
- `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_HISTORY`
- `FLAG_ACTIVITY_NO_USER_ACTION`
- `FLAG_ACTIVITY_PREVIOUS_IS_TOP`
- `FLAG_ACTIVITY_RESET_TASK_IF_NEEDED`
- `FLAG_ACTIVITY_SINGLE_TOP`
- `FLAG_RECEIVER_REGISTERED_ONLY`

**public Intent setType (String type)**

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 by `setData(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, and any MIME types you receive from outside of Android should be converted to lower case before supplying them here.

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)
- setDataAndType(Uri, String)

public String toString ()

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 simply concatenates the class name, the '@' sign and a hexadecimal representation of the object's hashCode(), that is, it is equivalent to the following expression:

```
geClass().getName() + '@' + Integer.toHexString(hashCode())
```

Returns

- a printable representation of this object.

public String toURI ()

public void writeToParcel (Parcel out, int flags)

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.