Intents and Intent Filters

An [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) is a messaging object you can use to request an action from another [app component](file:///G:\Studio\SDK\docs\guide\components\fundamentals.html#Components). Although intents facilitate communication between components in several ways, there are three fundamental use-cases（用例）:

* **To start an activity:**

An [Activity](file:///G:\Studio\SDK\docs\reference\android\app\Activity.html) represents a single screen in an app. You can start a new instance of an [Activity](file:///G:\Studio\SDK\docs\reference\android\app\Activity.html) by passing an[Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) to [startActivity()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Context.html" \l "startActivity(android.content.Intent)). The [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) describes the activity to start and carries any necessary data.

If you want to receive a result from the activity when it finishes, call [startActivityForResult()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\app\\Activity.html" \l "startActivityForResult(android.content.Intent, int)). Your activity receives the result as a separate [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) object in your activity's [onActivityResult()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\app\\Activity.html" \l "onActivityResult(int, int, android.content.Intent)) callback. For more information, see the [Activities](file:///G:\Studio\SDK\docs\guide\components\activities.html) guide.

* **To start a service:**

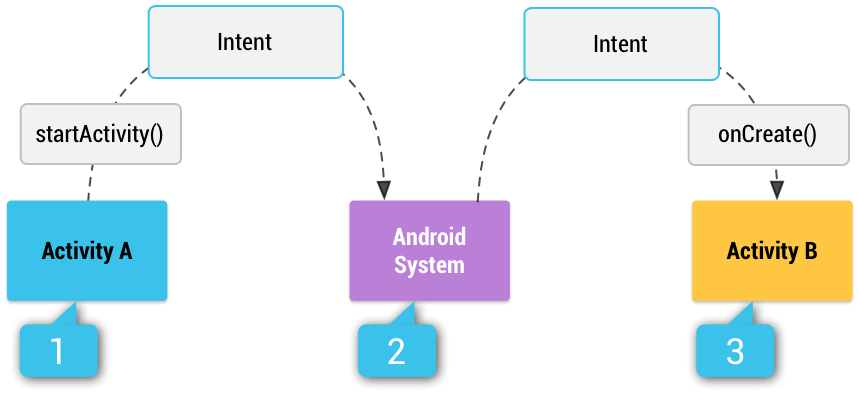
A [Service](file:///G:\Studio\SDK\docs\reference\android\app\Service.html) is a component that performs operations in the background without a user interface. You can start a service to perform a one-time operation (such as download a file) by passing an [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) to [startService()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Context.html" \l "startService(android.content.Intent)). The[Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) describes the service to start and carries any necessary data.

If the service is designed with a client-server interface, you can bind to the service from another component by passing an [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) to [bindService()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Context.html" \l "bindService(android.content.Intent, android.content.ServiceConnection, int)). For more information, see the[Services](file:///G:\Studio\SDK\docs\guide\components\services.html) guide.

* **To deliver a broadcast:**

A broadcast is a message that any app can receive. The system delivers various broadcasts for system events, such as when the system boots up or the device starts charging. You can deliver a broadcast to other apps by passing an [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) to [sendBroadcast()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Context.html" \l "sendBroadcast(android.content.Intent)), [sendOrderedBroadcast()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Context.html" \l "sendOrderedBroadcast(android.content.Intent, java.lang.String)), or[sendStickyBroadcast()](file:///G:\Studio\SDK\docs\reference\android\content\Context.html#sendStickyBroadcast(android.content.Intent)).

## Intent Types



**Figure 1.** Illustration of how an implicit intent is delivered through the system to start another activity: **[1]** Activity A creates an [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) with an action description and passes it to[startActivity()](file:///G:\Studio\SDK\docs\reference\android\content\Context.html#startActivity(android.content.Intent)). **[2]** The Android System searches all apps for an intent filter that matches the intent. When a match is found, **[3]** the system starts the matching activity (Activity B) by invoking its[onCreate()](file:///G:\Studio\SDK\docs\reference\android\app\Activity.html#onCreate(android.os.Bundle)) method and passing it the [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html).

There are two types of intents:

* **Explicit intents** specify the component to start by name (the fully-qualified class name). You'll typically use an explicit intent to start a component in your own app, because you know the class name of the activity or service you want to start. For example, start a new activity in response to a user action or start a service to download a file in the background.
* **Implicit intents** do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it. For example, if you want to show the user a location on a map, you can use an implicit intent to request that another capable app show a specified location on a map.

When you create an explicit intent to start an activity or service, the system immediately starts the app component specified in the [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) object.

When you create an implicit intent, the Android system finds the appropriate component to start by comparing the contents of the intent to the intent filters declared in the [manifest file](file:///G:\Studio\SDK\docs\guide\topics\manifest\manifest-intro.html) of other apps on the device. If the intent matches an intent filter, the system starts that component and delivers it the [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) object. If multiple intent filters are compatible, the system displays a dialog so the user can pick which app to use.

An intent filter is an expression in an app's manifest file that specifies the type of intents that the component would like to receive. For instance, by declaring an intent filter for an activity, you make it possible for other apps to directly start your activity with a certain kind of intent. Likewise, if you do not declare any intent filters for an activity, then it can be started only with an explicit intent.

**Caution:** To ensure your app is secure, always use an explicit intent when starting a [Service](file:///G:\\Studio\\SDK\\docs\\reference\\android\\app\\Service.html)and do not declare intent filters for your services. Using an implicit intent to start a service is a security hazard（冒险） because you cannot be certain what service will respond to the intent, and the user cannot see which service starts. Beginning with Android 5.0 (API level 21), the system throws an exception if you call [bindService()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Context.html" \l "bindService(android.content.Intent, android.content.ServiceConnection, int)) with an implicit intent.

## Building an Intent

An [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) object carries information that the Android system uses to determine which component to start (such as the exact component name or component category that should receive the intent), plus information that the recipient component uses in order to properly perform the action (such as the action to take and the data to act upon).

The primary information contained in an [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) is the following:

**Component name**

The name of the component to start.

This is optional, but it's the critical piece of information that makes an intent **explicit**, meaning that the intent should be delivered only to the app component defined by the component name. Without a component name, the intent is **implicit** and the system decides which component should receive the intent based on the other intent information (such as the **action, data, and category**—described below). So if you need to start a specific component in your app, you should specify the component name.

**Note:** When starting a [Service](file:///G:\Studio\SDK\docs\reference\android\app\Service.html), you should **always specify the component name**. Otherwise, you cannot be certain what service will respond to the intent, and the user cannot see which service starts.

This field of the [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) is a [ComponentName](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\ComponentName.html) object, which you can specify using a fully qualified class name of the target component, including the package name of the app. For example, com.example.ExampleActivity. You can set the component name with [setComponent()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Intent.html" \l "setComponent(android.content.ComponentName)),[setClass()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Intent.html" \l "setClass(android.content.Context, java.lang.Class<?>)), [setClassName()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Intent.html" \l "setClassName(java.lang.String, java.lang.String)), or with the [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) constructor.

**Action**

A **string** that specifies the generic action to perform (such as view or pick).

In the case of a broadcast intent, this is the action that took place and is being reported. **The action largely determines how the rest of the intent is structured—particularly what is contained in the data and extras**.

You can specify your own actions for use by intents within your app (or for use by other apps to invoke components in your app), but you should usually use action constants defined by the[Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) class or other framework classes. Here are some common actions for starting an activity:

[ACTION\_VIEW](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html#ACTION_VIEW)

Use this action in an intent with [startActivity()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Context.html" \l "startActivity(android.content.Intent)) when you have some information that an activity can show to the user, such as a photo to view in a gallery app, or an address to view in a map app.

[ACTION\_SEND](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html#ACTION_SEND)

Also known as the "share" intent, you should use this in an intent with[startActivity()](file:///G:\Studio\SDK\docs\reference\android\content\Context.html#startActivity(android.content.Intent)) when you have some data that the user can share through another app, such as an email app or social sharing app.

See the [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) class reference for more constants that define generic actions. Other actions are defined elsewhere in the Android framework, such as in [Settings](file:///G:\Studio\SDK\docs\reference\android\provider\Settings.html) for actions that open specific screens in the system's Settings app.

You can specify the action for an intent with [setAction()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Intent.html" \l "setAction(java.lang.String)) or with an [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) constructor.

If you define your own actions, be sure to include your app's package name as a prefix. For example:

static final String ACTION\_TIMETRAVEL = "com.example.action.TIMETRAVEL";

**Data**

The URI **(a**[**Uri**](file:///G:\Studio\SDK\docs\reference\android\net\Uri.html)**object)** that references the data to be acted on and/or the **MIME type** of that data. The type of data supplied is generally dictated by the intent's action. For example, if the action is [ACTION\_EDIT](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html#ACTION_EDIT), the data should contain the URI of the document to edit.

When creating an intent, it's often important to specify the type of data (its MIME type) in addition to its URI. For example, an activity that's able to display images probably won't be able to play an audio file, even though the URI formats could be similar. So specifying the MIME type of your data helps the Android system find the best component to receive your intent. However, the MIME type can sometimes be inferred from the URI—particularly when the data is a content: URI, which indicates the data is located on the device and controlled by a [ContentProvider](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\ContentProvider.html), which makes the data MIME type visible to the system.

To set only the data URI, call [setData()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Intent.html" \l "setData(android.net.Uri)). To set only the MIME type, call [setType()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Intent.html" \l "setType(java.lang.String)). If necessary, you can set both explicitly with [setDataAndType()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Intent.html" \l "setDataAndType(android.net.Uri, java.lang.String)).

**Caution:** If you want to set both the URI and MIME type, **do not** call [setData()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Intent.html" \l "setData(android.net.Uri)) and[setType()](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html#setType(java.lang.String)) because they each nullify the value of the other. Always use [setDataAndType()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Intent.html" \l "setDataAndType(android.net.Uri, java.lang.String))to set both URI and MIME type.

**Category**

A string containing additional information about the kind of component that should handle the intent. Any number of category descriptions can be placed in an intent, but most intents do not require a category. Here are some common categories:

[CATEGORY\_BROWSABLE](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html#CATEGORY_BROWSABLE)

The target activity allows itself to be started by a web browser to display data referenced by a link—such as an image or an e-mail message.

[CATEGORY\_LAUNCHER](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html#CATEGORY_LAUNCHER)

The activity is the initial activity of a task and is listed in the system's application launcher.

See the [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) class description for the full list of categories.

You can specify a category with [addCategory()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Intent.html" \l "addCategory(java.lang.String)).

These properties listed above (component name, action, data, and category) represent the defining characteristics of an intent. By reading these properties, the Android system is able to resolve which app component it should start.

However, an intent can carry additional information that does not affect how it is resolved to an app component. An intent can also supply:

**Extras**

Key-value pairs that carry additional information required to accomplish the requested action. Just as some actions use particular kinds of data URIs, some actions also use particular extras.

You can add extra data with various [putExtra()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Intent.html" \l "putExtra(java.lang.String, android.os.Bundle)) methods, each accepting two parameters: the key name and the value. You can also create a [Bundle](file:///G:\Studio\SDK\docs\reference\android\os\Bundle.html) object with all the extra data, then insert the [Bundle](file:///G:\Studio\SDK\docs\reference\android\os\Bundle.html) in the [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) with [putExtras()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Intent.html" \l "putExtras(android.content.Intent)).

For example, when creating an intent to send an email with [ACTION\_SEND](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html#ACTION_SEND), you can specify the "to" recipient with the [EXTRA\_EMAIL](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html#EXTRA_EMAIL) key, and specify the "subject" with the [EXTRA\_SUBJECT](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Intent.html" \l "EXTRA_SUBJECT)key.

The [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) class specifies many EXTRA\_\* constants for standardized data types. If you need to declare your own extra keys (for intents that your app receives), be sure to include your app's package name as a prefix. For example:

static final String EXTRA\_GIGAWATTS = "com.example.EXTRA\_GIGAWATTS";

**Flags**

Flags defined in the [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) class that function as metadata for the intent. The flags may instruct the Android system how to launch an activity (for example, which [task](file:///G:\Studio\SDK\docs\guide\components\tasks-and-back-stack.html) the activity should belong to) and how to treat it after it's launched (for example, whether it belongs in the list of recent activities).

For more information, see the [setFlags()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Intent.html" \l "setFlags(int)) method.

### Example explicit intent

An explicit intent is one that you use to launch a specific app component, such as a particular activity or service in your app. To create an explicit intent, define the component name for the[Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) object—all other intent properties are optional.

For example, if you built a service in your app, named DownloadService, designed to download a file from the web, you can start it with the following code:

// Executed in an Activity, so 'this' is the [Context](file:///G:\Studio\SDK\docs\reference\android\content\Context.html)  
// The fileUrl is a string URL, such as "http://www.example.com/image.png"  
Intent downloadIntent = new Intent(this, DownloadService.class);  
downloadIntent.setData([Uri.parse](file:///G:\\Studio\\SDK\\docs\\reference\\android\\net\\Uri.html" \l "parse(java.lang.String))(fileUrl));  
startService(downloadIntent);

The [Intent(Context, Class)](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html#Intent(android.content.Context, java.lang.Class<?>)) constructor supplies the app [Context](file:///G:\Studio\SDK\docs\reference\android\content\Context.html) and the component a [Class](file:///G:\\Studio\\SDK\\docs\\reference\\java\\lang\\Class.html)object. As such, this intent explicitly starts the DownloadService class in the app.

For more information about building and starting a service, see the [Services](file:///G:\Studio\SDK\docs\guide\components\services.html) guide.

### Example implicit intent

An implicit intent specifies an action that can invoke any app on the device able to perform the action. Using an implicit intent is useful when your app cannot perform the action, but other apps probably can and you'd like the user to pick which app to use.

For example, if you have content you want the user to share with other people, create an intent with the [ACTION\_SEND](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html#ACTION_SEND) action and add extras that specify the content to share. When you call[startActivity()](file:///G:\Studio\SDK\docs\reference\android\content\Context.html#startActivity(android.content.Intent)) with that intent, the user can pick an app through which to share the content.

**Caution:** It's possible that a user won't have any apps that handle the implicit intent you send to [startActivity()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Context.html" \l "startActivity(android.content.Intent)). If that happens, the call will fail and your app will crash. To verify that an activity will receive the intent, call [resolveActivity()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Intent.html" \l "resolveActivity(android.content.pm.PackageManager)) on your [Intent](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html) object. If the result is non-null, then there is at least one app that can handle the intent and it's safe to call [startActivity()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Context.html" \l "startActivity(android.content.Intent)). If the result is null, you should not use the intent and, if possible, you should disable the feature that issues the intent.

// Create the text message with a string  
Intent sendIntent = new Intent();  
sendIntent.setAction(Intent.ACTION\_SEND);  
sendIntent.putExtra(Intent.EXTRA\_TEXT, textMessage);  
sendIntent.setType("text/plain");  
  
// Verify that the intent will resolve to an activity  
if (sendIntent.resolveActivity(getPackageManager()) != null) {  
    startActivity(sendIntent);  
}

**Note:** In this case, a URI is not used, but the intent's data type is declared to specify the content carried by the extras.

When [startActivity()](file:///G:\\Studio\\SDK\\docs\\reference\\android\\content\\Context.html" \l "startActivity(android.content.Intent)) is called, the system examines all of the installed apps to determine which ones can handle this kind of intent (an intent with the [ACTION\_SEND](file:///G:\Studio\SDK\docs\reference\android\content\Intent.html#ACTION_SEND) action and that carries "text/plain" data). If there's only one app that can handle it, that app opens immediately and is given the intent. If multiple activities accept the intent, the system displays a dialog so the user can pick which app to use..