android developers

# Android 2.2 Platform

*API Level:* **8**

Android 2.2 is a minor platform release including user features, developer features, API changes, and bug fixes. For information on developer features and API changes, see the Framework API section.

For developers, the Android 2.2 platform is available as a downloadable component for the Android SDK. The downloadable platform includes a fully compliant Android library and system image, as well as a set of emulator skins, sample applications, and more. The downloadable platform includes no external libraries.



To get started developing or testing against the Android 2.2 platform, use the Android SDK and AVD Manager tool to download the platform into your SDK. For more information, see Adding SDK Components. If you are new to Android, download the SDK Starter Package first.

## Platform Highlights

For a list of new user features and platform highlights, see the Android 2.2 Platform Highlights document.

## Revisions

The sections below provide notes about successive releases of the Android 2.2 platform component for the Android SDK, as denoted by revision number. To determine what revision(s) of the Android 2.2 platforms are installed in your SDK environment, refer to the "Installed Packages" listing in the Android SDK and AVD Manager.

▼ Android 2.2, Revision 3 *(July 2011)*

Dependencies:

Requires SDK Tools r12 or higher.

Notes:

Improvements to the platform's rendering library to support the visual layout editor in the ADT Eclipse plugin. This revision allows for more drawing features in ADT and fixes several bugs in the previous rendering library. It also unlocks several editor features that were added in ADT 12.

▶ Android 2.2, Revision 2 *(July 2010)*

▶ Android 2.2, Revision 1 *(May 2010)*

## API Level

The Android 2.2 platform delivers an updated version of the framework API. The Android 2.2 API is assigned an integer

identifier — **8** — that is stored in the system itself. This identifier, called the "API Level", allows the system to correctly determine whether an application is compatible with the system, prior to installing the application.

To use APIs introduced in Android 2.2 in your application, you need to set the proper value, "8", in the `android:minSdkVersion` attributes of the `<uses-sdk>` element in your application's manifest.

For more information about how to use API Level, see the API Levels document.

---

# Framework API Changes

The sections below provide information about changes made to the application framework API provided by the Android 2.2 platform.

## App installation on external storage media

The Android platform now allows applications to request installation onto the device's external storage media (such as the SD card), as an alternative to installation onto the device's internal memory.

Application developers can express the preferred installation location for their applications by means of a new attribute of `<manifest>` in the manifest file, `android:installLocation`. The attribute supports three values: `"internalOnly"`, `"preferExternal"`, and `"auto"`. At install time, the system checks the value of `android:installLocation` and installs the application `.apk` according to the preferred location, if possible. If the application has requested external installation, the system installs it into a private, encrypted partition in the external media. Once an application .apk is installed externally, the system lets the user change the storage location of the .apk and move it onto the device's internal memory if needed (and vice versa), through Manage Applications in the user settings.

By default, the system installs all applications onto the device's internal memory, except for those that explicitly request external installation. This means that the system will always install legacy applications onto internal memory, since they do not have access to the `android:installLocation` attribute. However, it is possible to configure and compile a legacy application such that it is installed internally on older versions of the platform and externally on Android 2.2 and later platforms, if necessary.

Note that requesting installation onto the device's external media is not suitable for all applications, particularly because the external media may be removable and unmounting/remounting may disrupt the user experience and system settings.

For more information about setting a preferred install location for your application, including a discussion of what types of applications should and should not request external installation, please read the App Install Location document.

## Data backup

The platform now provides a generalized backup service that applications can use to backup and restore user data, to ensure that users can maintain their data when switching devices or reinstalling the application. The Backup Manager handles the work of transporting the application data to and from the backup storage area in the cloud. The Backup Manager can store any type of data, from arbitrary data to files, and manages backup and restore operations in an atomic manner. For more information, see Data Backup.

## Graphics

- New OpenGL ES 2.0 APIs in `android.opengl.GLES20`.
- New `ETC1`, `ETC1Util`, and `ETC1Util.ETC1Texture` classes and utility methods for using ETC1 for texture compression.
- New `ImageFormat` class.
- New `YUV image format API` to enable compression from YUV to JPEG and manipulation of YUV data.

## Media

- New APIs in `android.media.AudioManager` for managing audio focus, transport control, transient loss of audio focus, ducking.
- New broadcast intent for routing audio to SCO — `ACTION_SCO_AUDIO_STATE_CHANGED` with extras indicating new state.
- New APIs in `SoundPool` to detect completion of sound-loading.
- New APIs in `SoundPool` for auto pause and resume.
- New APIs in `MediaRecorder` for specifying audio settings for number of channels, encoding and sampling rates, sampling rate.
- New APIs for adding files to the media database, so that they are automatically scanned. See `MediaScannerConnection.scanFile` and `MediaScannerConnection.OnScanCompletedListener`.

## Speech recognition and third-party recognition engines

- The platform provides new speech-recognition APIs that allow applications to have a richer interaction with the available voice recognizer. For example, the APIs are sufficient to integrate voice recognition deeply into an IME.
- The platform also provides a `RecognitionService` base class that lets third-party developers create plug-in recognition engines.
- New `RecognitionListener` interface to receive callbacks.
- New `RecognizerIntent` extras that let a requester app specify details as preferred language, minimum length in milliseconds, and so on.

## Camera and camcorder

- Changes to camera preview API to improve efficieny of preview pipeline.
- New display orientation for camera (it can now work in portrait orientation).
- New APIs in `android.hardware.Camera` for managing zoom level.
- New APIs `android.hardware.Camera.Parameters` for querying and setting device camera settings such as focal length, exposure, zoom level, view angle, and others.
- New `thumbnail` utility for video and image thumbnails.
- New `CamcorderProfile` and `CamcorderProfile` classes enable apps to determine device hardware camera capablities.
- New support in `android.media.ExifInterface` for retrieving GPS and focal length.

## Device policy manager

New device policy management APIs allow developers to write "device administrator" applications that can control security features of the device, such as the minimum password strength, data wipe, and so on. Users can select the administrators that are enabled on their devices. For more information, see the `android.app.admin` classees or the example application code in DeviceAdminSample.java.

## UI Framework

- New UI modes "car mode" and "night mode" and `UiModeManager` let applications adjust their application UI for specific user modes.
- New `ScaleGestureDetector` that lets Views detect and handle transformation gestures that involve more than one pointer (multitouch) using the supplied MotionEvents.
- Improvements in the way that multitouch events are reported in `MotionEvent` objects.
- The layout attribute `fill_parent` is renamed to `match_parent`. This affects both XML and Java code (see `ViewGroup.LayoutParams`). Note that the platform will continue to honor uses of `fill_parent` in legacy applications.
- New layout attributes `tabStripEnabled`, `tabStripRight`, and `tabStripLeft` let developers customize the

bottom strip of TabWidgets.

- Better support for managed dialogs in Activity.

### Accounts and sync

- New method `AddPeriodicSync()` lets you schedule a periodic sync with a specific account, authority, and extras at the given frequency.

### New manifest elements and attributes

- For specifying the application's preferred install location (see App Installation on External Storage Media, above):
  - New `android:installLocation` attribute of the `<manifest>` element. Specifies the default install location defined by an application.
- For managing user data backup (see Backup manager, above, for more information):
  - New `android:backupAgent` attribute of the `<application>` element. Specifies the component name of the BackupAgent subclass provided by the application to handle backup/restore operations, if any.
  - New `android:restoreAnyVersion` attribute of the `<application>` element. Boolean value that indicates whether the application is prepared to attempt a restore of any backed-up dataset, even if the backup is apparently from a newer version of the application than is currently installed on the device.
- For managing the platform's JIT compiler:
  - New `android:vmSafeMode` attribute of the `<application>` element. Boolean value that specifies whether to disable JIT compiler optimizations when running the application.

### Permissions

- `android.permission.BIND_DEVICE_ADMIN` — Any device administration broadcast receiver must require this permission, to ensure that only the system can interact with it.
- `android.permission.KILL_BACKGROUND_PROCESSES` — Allows an application to call `killBackgroundProcesses(String)`.
- `android.permission.BIND_WALLPAPER` — Any `WallpaperService` must require this permission, to ensure that only the system can interact with it.
- `android.permission.SET_TIME` — Allows an application to set the system time.

### API differences report

For a detailed view of all API changes in Android 2.2 (API Level 8), see the API Differences Report.

---

## Built-in Applications

The system image included in the downloadable platform provides these built-in applications:

- Alarm Clock
- Browser
- Calculator
- Camera
- Contacts
- Custom Locale (developer app)
- Dev Tools (developer app)
- Gallery
- IMEs for Japanese, Chinese, and Latin text input
- Messaging
- Music
- Phone
- Settings
- Spare Parts (developer app)

- Email

## Locales

The system image included in the downloadable platform provides a variety of built-in locales. In some cases, region-specific strings are available for the locales. In other cases, a default version of the language is used. The languages that are available in the Android 2.2 system image are listed below (with *language_country/region* locale descriptor).

- Chinese, PRC (zh_CN)
- Chinese, Taiwan (zh_TW)
- Czech (cs_CZ)
- Dutch, Netherlands (nl_NL)
- Dutch, Belgium (nl_BE)
- English, US (en_US)
- English, Britain (en_GB)
- English, Canada (en_CA)
- English, Australia (en_AU)
- English, New Zealand (en_NZ)
- English, Singapore(en_SG)
- French, France (fr_FR)
- French, Belgium (fr_BE)
- French, Canada (fr_CA)
- French, Switzerland (fr_CH)
- German, Germany (de_DE)
- German, Austria (de_AT)
- German, Switzerland (de_CH)
- German, Liechtenstein (de_LI)
- Italian, Italy (it_IT)
- Italian, Switzerland (it_CH)
- Japanese (ja_JP)
- Korean (ko_KR)
- Polish (pl_PL)
- Russian (ru_RU)
- Spanish (es_ES)

Localized UI strings match the locales that are accessible through Settings.

> **Note:** Android supports more locales than are listed above. However, the entire collection of locale strings cannot fit on a single system image, so the above list is only what's included in the system image for the SDK. All of Android's supported locales are available in the Android Open Source Project.

## Emulator Skins

The downloadable platform includes a set of emulator skins that you can use for modeling your application in different screen sizes and resolutions. The emulator skins are:

- QVGA (240x320, low density, small screen)
- WQVGA (240x400, low density, normal screen)
- FWQVGA (240x432, low density, normal screen)
- HVGA (320x480, medium density, normal screen)
- WVGA800 (480x800, high density, normal screen)
- WVGA854 (480x854 high density, normal screen)

For more information about how to develop an application that displays and functions properly on all Android-powered devices, see Supporting Multiple Screens.

↑ Go to top