Android Developers

Start the Emulator from the Command Line

The Android SDK includes an Android device emulator — a virtual device that runs on your computer. The Android Emulator lets you develop and test Android apps without using a physical device.

This page describes command-line features that you can use with the Android Emulator. For information about using the Android Emulator UI, see Run Apps on the Android Emulator (https://developer.android.com/studio/run/emulator.html).

Starting the emulator

Use the `emulator` command to start the emulator, as an alternative to running your project (https://developer.android.com/studio/run/emulator.html#runningapp) or starting it through the AVD Manager (https://developer.android.com/studio/run/emulator.html#runningemulator).

Here's the basic command-line syntax for starting a virtual device from a terminal prompt:

```
emulator -avd avd_name [ {-option [value]} … ]
```

Or

```
emulator @avd_name [ {-option [value]} … ]
```

For example, if you launch the emulator from within Android Studio running on a Mac, the default command line will be similar to the following:

```
$ /Users/janedoe/Library/Android/sdk/tools/emulator -avd Nexus_5X_API_23 -netd
```

You can specify startup options when you start the emulator, but not later on.

For a list of AVD names, enter the following command:

```
emulator -list-avds
```

When you use this option, it displays a list of AVD names from your Android home directory. Note that you can override the default home directory by setting the `ANDROID_SDK_HOME` environment variable: the root of the user-specific directory where all configuration and AVD content is stored. You could set the environment variable in the terminal window before launching a virtual device, or you could set it through your user settings in the operating system; for example, in your `.bashrc` file on Linux.

To stop the Android Emulator, just close the emulator window.

Installing an app

In addition to installing an app through Android Studio (https://developer.android.com/studio/run/emulator.html#runningapp) or the emulator UI (https://developer.android.com/studio/run/emulator.html#tasks), you can install your app on a virtual device by using the adb (https://developer.android.com/tools/help/adb.html#move) utility.

To install an app by using adb, and then run and test the app, follow these general steps:

1. Build and package your app into an APK as described in Build and Run Your App (https://developer.android.com/studio/run/index.html).

2. Start the emulator from the command line as described in the previous section, using any startup options necessary.

3. Install your app using adb (https://developer.android.com/tools/help/adb.html#move).

4. Run and test your app on the emulator.
   While the emulator is running, you can also use the Emulator Console (https://developer.android.com/studio/run/emulator-console.html) to issue commands as needed.

5. The virtual device preserves the app and its state data across restarts, in a user-data disk partition (`userdata-qemu.img`). To clear this data, start the emulator with the `-wipe-data` option or wipe the data in the AVD Manager, for example. For more information about the user-data partition and other storage, see the following section.
   To uninstall an app, do so as you would on an Android device.

> **Note:** The `adb` utility sees the virtual device as an actual physical device. For this reason, you might have to use the `-d` flag with some common `adb` commands, such as `install`. The `-d` flag lets you specify which of several connected devices to use as the target of a command. If you don't specify `-d`, the emulator targets the first device in its list.

Understanding the default directories and files

The emulator uses associated files, of which the AVD system and data directories are the most important. It helps to understand the emulator directory structure and files when specifying command-line options. Although, you normally don't need to modify the default directories or files.

The Android Emulator uses the Quick Emulator (QEMU (http://wiki.qemu.org/)   ) hypervisor. Initial versions of the Android Emulator used QEMU 1 (goldfish), and later versions use QEMU 2 (ranchu).

AVD system directory

The system directory contains the Android system images that the emulator uses to simulate the operating system. It has platform-specific, read-only files shared by all AVDs of the same type, including API level, CPU architecture, and Android variant. The default locations are the following:

- Mac OS X and Linux - `~/Library/Android/sdk/system-images/android-`*`apiLevel`*`/`*`variant`*`/`*`arch`*`/`

- Microsoft Windows XP - `C:\Documents and Settings\`*`user`*`\Library\Android\sdk\system-images\android-`*`apiLevel`*`\`*`variant`*`\`*`arch`*`\`

- Windows Vista - `C:\Users\`*`user`*`\Library\Android\sdk\system-images\android-`*`apiLevel`*`\`*`variant`*`\`*`arch`*`\`

Where:

- *`apiLevel`* is a numeric API level, or a letter for preview releases. For example, `android-M` indicated the Android Marshmallow preview. On release, it became API level 23, designated by `android-23`.

- *`variant`* is a name corresponding to specific features implemented by the system image; for example, `google_apis` or `android-wear`.

- *`arch`* is the target CPU architecture; for example, `x86`.

Use the `-sysdir` option to specify a different system directory for the AVD.

The emulator reads the following files from the system directory.

File	Description	Option to Specify a Different File
`kernel-qemu` or `kernel-ranchu`	The binary kernel image for the AVD. `kernel-ranchu` is the QEMU 2 emulator, the latest version.	`-kernel`
`system.img`	The read-only initial version of the system image; specifically, the partition containing the system libraries and data corresponding the API level and variant.	`-system`
`ramdisk.img`	The boot partition image. This is a subset of `system.img` that's loaded by the kernel initially before the system image is mounted. It typically contains just a few binaries and initialization scripts.	`-ramdisk`
`userdata.img`	The *initial* version of the data partition, which appears as `data/` in the emulated system and contains all writable data for the AVD. The emulator uses this file when you create a new AVD or use the `-wipe-data` option. For more information, see the `userdata-qemu.img` file description in the following section.	`-initdata` `-init-data`

AVD data directory

The AVD data directory, also called the content directory, is specific to a single AVD instance and contains all modifiable data for the AVD.

The default location is the following, where *`name`* is the AVD name:

- Mac OS X and Linux - `~/.android/avd/`*`name`*`.avd/`

- Microsoft Windows XP - `C:\Documents and Settings\`*`user`*`\.android\`*`name`*`.avd\`

- Windows Vista, and higher - `C:\Users\`*`user`*`\.android\`*`name`*`.avd\`

Use the `-datadir` option to specify a different AVD data directory.

The following table lists the most important files contained in this directory.

File	Description	Option to Specify a Different File

userdata-qemu.img	The content of the data partition, which appears as `data/` in the emulated system. When you create a new AVD, or when you use the `-wipe-data` option to reset the AVD to the factory defaults, the emulator copies the `userdata.img` file in the system directory to create this file. Each virtual device instance uses a writable user-data image to store user- and session-specific data. For example, it uses the image to store a unique user's installed app data, settings, databases, and files. Each user has a different `ANDROID_SDK_HOME` directory that stores the data directories for the AVDs created by that user; each AVD has a single `userdata-qemu.img` file.	`-data`
cache.img	The cache partition image, which appears as `cache/` in the emulated system. It's empty when you first create an AVD or use the `-wipe-data` option. It stores temporary download files and is populated by the download manager and sometimes the system; for example, the browser uses it to cache downloaded web pages and images while the emulator is running. When you power off the virtual device, the file is deleted. You can persist the file by using the `-cache` option.	`-cache`
sdcard.img	(Optional) An SD card partition image that lets you simulate an SD card on a virtual device. You can create an SD card image file in the AVD Manager (https://developer.android.com/studio/run/managing-avds.html) or using the `mksdcard` (https://developer.android.com/studio/command-line/mksdcard.html) tool. The file is stored on your development computer and must be loaded at startup. When defining an AVD in the AVD Manager, you have the choice to use an automatically managed SD card file, or a file that you created with the `mksdcard` tool. You can view the `sdcard.img` file associated with an AVD in the AVD Manager. The `-sdcard` option overrides the SD card file specified in the AVD. You can browse, send files to, and copy and remove files from a simulated SD card by using the emulator UI or the adb (https://developer.android.com/studio/command-line/adb.html#copyfiles) utility while the virtual device is running. You can't remove a simulated SD card from a running virtual device. To copy files to the SD card file before loading it, you can mount the image file as a loop device and then copy the files. Or use a utility such as the `mtools` package to copy the files directly to the image. The emulator treats the file as a pool of bytes so the SD card format doesn't matter. Note that the `-wipe-data` option doesn't affect this file. If you want to clear the file, you need to delete the file and then recreate it using the AVD Manager or the `mksdcard` tool. Changing the size of the file also deletes the file and creates a new file.	`-sdcard`

Listing directories and files used by the emulator

You can discover where files are located in two ways:

- When you start the emulator from the command line, use the `-verbose` or `-debug init` option, and look at the output.

- Use the `emulator -help-`*option* command to list a default directory. For example:

```
$ emulator -help-datadir

  Use '-datadir <dir>' to specify a directory where writable image files
  will be searched. On this system, the default directory is:

      /Users/me/.android

  See '-help-disk-images' for more information about disk image files.
```

Command-line startup options

This section lists options you can supply on the command line when you start the emulator.

Note: The Android Emulator is continually under development to make it more reliable. For status on the issues reported against various command-line options, and to report bugs, see the Android Issue Tracker (https://code.google.com/p/android/issues/list?can=2&q=%22emulator%22%20%20Subcomponent%3DTools-emulator%20opened-after%3A2016/1/1&colspec=ID%20Status%20Priority%20Owner%20Summary%20Stars%20Reporter%20Opened&sort=-id&num=100&start=100) .

Commonly used options

The following table lists command-line startup options that you might use more often.

Command-Line Option	Description
Device Hardware	

`-camera-back` *`mode`*
`-camera-front` *`mode`*
	Set the emulation mode for a camera facing back or front. It overrides any camera setting in the AVD. *`mode`* can be any of the following values:
	
	• `emulated` - The emulator simulates a camera in the software.
	
	• `webcam`*`n`* - The emulator uses a webcam connected to your development computer. For a list of webcams, use the `-webcam-list` option; for example, `webcam0`.
	
	• `none` - Disable the camera in the virtual device.
	
	For example:
	
	``
	$ emulator @Nexus_5X_API_23 -camera-back webcam0
	``

`-webcam-list`	List the web cameras on your development computer that are available for emulation. For example:
	
	``
	$ emulator @Nexus_5X_API_23 -webcam-list
	List of web cameras connected to the computer:
	  Camera 'webcam0' is connected to device 'webcam0' on channel 0 using
	  pixel format 'UYVY'
	``
	
	In the example, the first `webcam0` is the name you use on the command line. The second `webcam0` is the name used by the OS on the development computer. The second name varies depending on the OS.
	
	As of SDK Tools 25.2.4, the AVD name is required, although it might not be in the future.

Disk Images and Memory	

`-memory` *`size`*	Specify the physical RAM size from 128 to 4096 MBs. For example:
	
	``
	$ emulator @Nexus_5X_API_23 -memory 2048
	``
	
	This value overrides the AVD setting.

`-sdcard` *`filepath`*	Specify the filename and path to an SD card partition image file. For example:
	
	``
	$ emulator @Nexus_5X_API_23 -sdcard C:/sd/sdcard.img
	``
	
	If the file isn't found, the emulator still launches, but without an SD card; the command returns a **No SD Card Image** warning.
	
	If you don't specify this option, the default is `sdcard.img` in the data directory (unless the AVD specifies something different). For details about emulated SD cards, see AVD data directory (#data-filedir).

`-wipe-data`	Delete user data and copy data from the initial data file. This option clears the data for the virtual device and returns it to the same state as when it was first defined. All installed apps and settings are removed. For example:
	
	``
	$ emulator @Nexus_5X_API_23 -wipe-data
	``
	
	By default, the user data file is `userdata-qemu.img` and the initial data file is `userdata.img`, both residing in the data directory. The `-wipe-data` option doesn't affect the `sdcard.img` file. For more information about user data, see Understanding the default directories and files (#filedir).

Debug	

`-debug` *`tags`*	Enable or disable the display of debug messages for one or more tags. Separate multiple tags by a space, comma, or column. For example:
	
	``
	$ emulator @Nexus_5X_API_23 -debug init,metrics
	``
	
	To disable a tag, place a dash (-) in front of it; for example, the following option displays all debug messages, except the ones related to network sockets and metrics:
	
	`-debug all,-socket,-metrics`
	
	For a list of tags and descriptions, use the `-help-debug-tags` option. For example:
	
	``
	$ emulator -help-debug-tags
	``
	
	You can define the default debug tags in the `ANDROID_VERBOSE` (https://developer.android.com/studio/command-line/variables.html#android_verbose) environment variable. Define the tags you want to use in a comma-delimited list. Here's an example showing it defined with the `socket` and `gles` tags:

```
ANDROID_VERBOSE=socket,gles
```

It's equivalent to using:

```
-debug-socket -debug-gles
```

`-debug-`*`tag`* `-debug-no-`*`tag`*	Enable a specific debug message type. Use the `no` form to disable a debug message type. For example: ```$ emulator @Nexus_5X_API_23 -debug-all -debug-no-metrics``` For a list of tags, use the `emulator -help-debug-tags` command.

`-logcat` *`logtags`*	Enable the display of logcat messages for one or more tags, and write them to the terminal window. For example, the following command enables error messages from all components:
	
	```$ emulator @Nexus_5X_API_23 -logcat *:e```
	
	`logtags` uses the same format as the `adb logcat` *`logtags`* command (enter `adb logcat -help` for more information). It's a list of space- or comma-separated log filters of the format *`componentName`*`:`*`logLevel`*. *`componentName`* is either a wildcard asterisk (`*`) or a component name, such as ActivityManager, SystemServer, InputManager, WindowManager, and so on. *`logLevel`* is one of these values:
	
	• `v` - verbose
	
	• `d` - debug
	
	• `i` - informative
	
	• `w` - warning log level
	
	• `e` - error
	
	• `s` - silent
	
	The following example displays GSM component messages at the informative log level:
	
	```$ emulator @Nexus_5X_API_23 -logcat '*:s GSM:i'```
	
	If you don't supply the `-logcat` option on the command line, the emulator looks for the `ANDROID_LOG_TAGS` (https://developer.android.com/studio/command-line/variables.html#android_log_tags) environment variable. If `ANDROID_LOG_TAGS` is defined with a valid *`logtags`* value and isn't empty, the emulator uses its value to enable logcat output to the terminal by default. You can also redirect the same, or other, log messages to the terminal through adb. For more information about logcat and adb, see logcat Command-Line Tool (https://developer.android.com/studio/command-line/logcat.html), Write and View Logs with Logcat (https://developer.android.com/studio/debug/am-logcat.html), Log (https://developer.android.com/reference/android/util/Log.html) class, and adb commands reference (https://developer.android.com/studio/command-line/adb.html#issuingcommands).

`-show-kernel`	Display kernel debug messages in the terminal window. For example: ```$ emulator @Nexus_5X_API_23 -show-kernel``` One use of this option is to check that the boot process works correctly.
`-verbose`	Print emulator initialization messages to the terminal window. For example: ```$ emulator @Nexus_5X_API_23 -verbose``` It displays which files and settings are actually selected when starting a virtual device defined in an AVD. This option is the same as specifying `-debug-init`.
Network	
`-dns-server` *`servers`*	Use the specified DNS servers. *`servers`* is a comma-separated list of up to four DNS server names or IP addresses. For example: ```$ emulator @Nexus_5X_API_23 -dns-server 192.0.2.0,192.0.2.255``` By default, the emulator tries to detect the DNS servers you're using and sets up special aliases in the emulated firewall network to allow the Android system to connect directly to them. Use the `-dns-server` option to specify a different list of DNS servers.
`-http-proxy` *`proxy`*	Make all TCP connections through a specified HTTP/HTTPS proxy. If your emulator must access the internet through a proxy server, you can use this option or the `http_proxy` environment variable to set up the appropriate redirection. For example:

```
$ emulator @Nexus_5X_API_23 -http-proxy myserver:1981
```

proxy can be one of the following:

`http://`*server*`:`*port*

`http://`*username*`:`*password*`@`*server*`:`*port*

The `http://` prefix can be omitted.

If this option isn't supplied, the emulator looks up the `http_proxy` environment variable and automatically uses any value matching the *proxy* format. For more information, see Using the emulator with a proxy (https://developer.android.com/studio/run/emulator-networking.html#proxy).

`-netdelay` *delay*	Set network latency emulation to one of the following *delay* values in milliseconds:

- `gsm` - GSM/CSD (min 150, max 550).

- `hscsd` - HSCSD (min 80, max 400).

- `gprs` - GPRS (min 35, max 200).

- `edge` - EDGE/EGPRS (min 80, max 400).

- `umts` - UMTS/3G (min 35, max 200).

- `hsdpa` - HSDPA (min 0, max 0).

- `lte` - LTE (min 0, max 0).

- `evdo` - EVDO (min 0, max 0).

- `none` - No latency, the default (min 0, max 0).

- *num* - Specify exact latency.

- *min*`:`*max* - Specify individual minimum and maximum latencies.

For example:

```
$ emulator @Nexus_5X_API_23 -netdelay gsm
```

The emulator supports network throttling (limiting the maximum network bandwidth, also called network shaping) as well as higher connection latencies. You can define it either through the skin configuration, or with the `-netspeed` and `-netdelay` options.

`-netfast`	Disable network throttling. For example:

```
$ emulator @Nexus_5X_API_23 -netfast
```

This option is the same as specifying `-netspeed full -netdelay none`. These are the default values for these options.

`-netspeed` *speed*	Set the network speed emulation. Specify the maximum network upload and download speeds with one of the following *speed* values in kbps:

- `gsm` - GSM/CSD (up: 14.4, down: 14.4).

- `hscsd` - HSCSD (up: 14.4, down: 57.6).

- `gprs` - GPRS (up: 28.8, down: 57.6).

- `edge` - EDGE/EGPRS (up: 473.6, down: 473.6).

- `umts` - UMTS/3G (up: 384.0, down: 384.0).

- `hsdpa` - HSDPA (up: 5760.0, down: 13,980.0).

- `lte` - LTE (up: 58,000, down: 173,000).

- `evdo` - EVDO (up: 75,000, down: 280,000).

- `full` - No limit, the default (up: 0.0, down: 0.0).

- *num* - Specify both upload and download speed.

- *up*`:`*down* - Specify individual up and down speeds.

For example:

```
$ emulator @Nexus_5X_API_23 -netspeed edge
```

The emulator supports network throttling (limiting the maximum network bandwidth, also called network shaping) as well as higher connection latencies. You can define it either through the skin configuration,

	or with the `-netspeed` and `-netdelay` options.
`-port` *port*	Set the TCP port number that's used for the console and adb. For example: ```$ emulator @Nexus_5X_API_23 -port 5556``` The default value is 5554 for the first virtual device instance running on the your machine. A virtual device normally occupies a pair of adjacent ports: a console port and an adb port. The console of the first virtual device running on a particular machine uses console port 5554 and adb port 5555. Subsequent instances use port numbers increasing by two — for example, 5556/5557, 5558/5559, and so on. The range is 5554 to 5682, allowing for 64 concurrent virtual devices. The port assignments are often the same as specifying `-ports` *port*,{*port* + 1}. {*port* + 1} must be free and will be reserved for adb. If any of the console or adb ports is already in use, the emulator won't start. The `-port` option reports which ports and serial number the virtual device is using, and warns if there are any issues with the values you provided. In the emulator UI, you can see the console port number in the window title, and you can view the adb port number by selecting **Help** > **About**. Note that if the *port* value is not even and is in the range 5554 to 5584, the virtual device will start but not be visible when you use the `adb devices` command if the adb server starts *after* the emulator. For this reason, we recommend using an even console port number.
`-ports` *console-port,adb-port*	Set the TCP ports used for the console and adb. For example: ```$ emulator @Nexus_5X_API_23 -ports 5556,5559``` The valid ports range is 5554 to 5682, allowing for 64 concurrent virtual devices. The `-ports` option reports which ports and serial number the emulator instance is using, and warns if there are any issues with the values you provided. We recommend using the `-port` option instead, where possible. The `-ports` option is available for network configurations that require special settings. For more information about setting console and adb ports, see the `-port` option.
`-tcpdump` *filepath*	Capture network packets and store them in a file. For example: ```$ emulator @Nexus_5X_API_23 -tcpdump /path/dumpfile.cap``` Use the option to begin capturing all network packets that are sent through the virtual Ethernet LAN of the emulator. After, you can use a tool like Wireshark to analyze the traffic. Note that this option captures all Ethernet packets, and isn't limited to TCP connections.
System	
`-accel` *mode*	Configure emulator VM acceleration. For example: ```$ emulator @Nexus_5X_API_23 -accel auto``` Accelerated emulation works for x86 and x86_64 system images only. On Linux, it relies on KVM. On Windows and Mac, it relies on an Intel CPU and Intel HAXM driver. This option is ignored if you're not emulating an x86 or x86_64 device. Valid values for *mode* are: • `auto` - Determine automatically if acceleration is supported and use it when possible (default). • `off` - Disables acceleration entirely, which is primarily useful for debugging. • `on` - Force acceleration. If KVM or HAXM isn't installed or usable, the emulator won't start and prints an error message. For more information, see Configure Hardware Acceleration (https://developer.android.com/studio/run/emulator-acceleration.html).
`-accel-check`	Check whether a required hypervisor for emulator VM acceleration is installed (HAXM or KVM). For example: ```$ emulator -accel-check``` For more information, see Determining whether HAXM or KVM is installed (https://developer.android.com/studio/run/emulator-acceleration.html#accel-check).
`-engine` *engine*	Specify the emulator engine: • `auto` - Automatically select an engine (default).

- `classic` - Use the older QEMU 1 engine.

- `qemu2` - Use the newer QEMU 2 engine.

For example:

```
$ emulator @Nexus_5X_API_23 -engine auto
```

Auto-detection should choose the value that provides the best performance when emulating a particular AVD. You should use the `-engine` option for debugging and comparison purposes only.

`-gpu` *`mode`*	Select the GPU emulation mode. For example: ```$ emulator @Nexus_5X_API_23 -gpu swiftshader``` For more information, see Configuring graphics acceleration on the command line (https://developer.android.com/studio/run/emulator-acceleration.html#accel-graphics).
`-no-accel`	Disable emulator VM acceleration when using an x86 or x86_64 system image. It's useful for debugging only and is the same as specifying `-accel off`. For example: ```$ emulator @Nexus_5X_API_23 -no-accel``` For more information, see Configure Hardware Acceleration (https://developer.android.com/studio/run/emulator-acceleration.html).
`-nojni` `-no-jni`	Disable extended Java Native Interface (JNI) checks in the Android Dalvik or ART runtime. For example: ```$ emulator @Nexus_5X_API_23 -nojni``` When you start a virtual device, extended JNI checks are enabled by default. For more information, see JNI Tips (https://developer.android.com/training/articles/perf-jni.html).
`-selinux` `{disabled|permissive}`	Set the Security-Enhanced Linux (SELinux (https://en.wikipedia.org/wiki/Security-Enhanced_Linux)) security module to either disabled or permissive mode on a Linux operating system. For example: ```me-linux$ emulator @Nexus_5X_API_23 -selinux permissive``` By default, SELinux is in enforcing mode, meaning the security policy is enforced. `permissive` mode loads the SELinux policy, but doesn't enforce it; it just logs policy violations. `disabled` mode disables kernel support for SELinux.
`-timezone` *`timezone`*	Set the timezone for the virtual device to *`timezone`*, instead of the host timezone. For example: ```$ emulator @Nexus_5X_API_23 -timezone Europe/Paris``` By default, the emulator uses the timezone of your development computer. Use this option to specify a different timezone or if the automatic detection isn't working correctly. The *`timezone`* value must be in zoneinfo (https://en.wikipedia.org/wiki/List_of_tz_database_time_zones) format, which is *`area`*/*`location`* or *`area`*/*`subarea`*/*`location`*. For example: - `America/Los_Angeles` - `Europe/Paris` - `America/Argentina/Buenos_Aires` The specified timezone must be in the zoneinfo database (https://www.iana.org/time-zones).

`-version`	Display the emulator version number. For example:
	
	```$ emulator @Nexus_5X_API_23 -version```
	
	Or
	
	```$ emulator -version```

UI	
`-no-boot-anim`	Disable the boot animation during emulator startup for faster booting. For example: ```$ emulator @Nexus_5X_API_23 -no-boot-anim``` On slower computers, this option can significantly speed up the boot sequence.

-screen *mode*	Set emulated touch screen mode. For example: ```$ emulator @Nexus_5X_API_23 -screen no-touch``` *mode* can be any of the following values: • touch - Emulate a touch screen (default). • multi-touch - Emulate a multi-touch screen. • no-touch - Disable touch and multi-touch screen emulation.

Advanced options

The following command-line startup options are available, but not commonly used by the average app developer.

In the descriptions, the *working directory* is the current directory in the terminal where you're entering commands. For information about the AVD *system directory* and *data directory*, and the files stored within them, see Understanding the default directories and files (#filedir).

Some of these options are appropriate for external app developers, and some of them are used primarily by platform developers. *App developers* create Android apps and run them on specific AVDs. *Platform developers* work on the Android system and run it inside the emulator with no pre-created AVD; they're internal Android team members, not external app developers.

Advanced Option	Brief Description
-bootchart *timeout*	Enable bootcharting, with a timeout in seconds. Some Android system images have a modified init system that integrates a bootcharting facility (http://www.bootchart.org/) . You can pass a bootcharting timeout period to the system with this option. If your init system doesn't have bootcharting activated, the option does nothing. This option is primarily useful to platform developers, not external app developers. For example: ```$ emulator @Nexus_5X_API_23 -bootchart 120```
-cache *filepath*	Specify a cache partition image file. Provide a filename, and an absolute path or a path relative to the data directory, to set up a persistent cache file. If the file doesn't exist, the emulator creates it as an empty file. If you don't use this option, the default is a temporary file named cache.img. For more information, see AVD data directory (#data-filedir). For example: ```$ emulator @Nexus_5X_API_23 -cache ~/.android/avd/Nexus_5X_API_23.avd/cache_persistent.img```
-cache-size *size*	Set the cache partition size in MBs. If you don't specify this option, the default is 66 MB. Normally, most app developers don't need this option, unless they need to download very large files that are larger than the default cache. For more information about the cache file, see AVD data directory (#data-filedir). For example: ```$ emulator @Nexus_5X_API_23 -cache-size 1000```
-data *filepath*	Set the user data partition image file. Provide a filename, and an absolute path or a path relative to the working directory, to set up a persistent user data file. If the file doesn't exist, the emulator creates an image from the default userdata.img file, stores it in the filename you specified, and persists user data to it at shutdown. If you don't use this option, the default is a file named userdata-qemu.img. For more information about the user data file, see AVD data directory (#data-filedir). For example: ```$ emulator @Nexus_5X_API_23 -data ~/.android/avd/Nexus_5X_API_23.avd/userdata-test.img```
-datadir *dir*	Specify a data directory using an absolute path. For more information, see AVD data directory (#data-filedir). For example: ```$ emulator @Nexus_5X_API_23 -datadir ~/.android/avd/Nexus_5X_API_23.avd/mytest```
-force-32bit	Use the 32-bit emulator on 64-bit platforms. Occasionally, this option is useful for testing or debugging. For example, there was an issue where the emulator would sometimes not run on 64-bit Windows, but 32-bit did run; this option was helpful for performing comparisons to debug the issue. Here's an example: ```$ emulator @Nexus_5X_API_23 -force-32bit```

-help-disk-images	Get help about about disk images. It provides information relevant to both app and platform developers. For example: ```$ emulator -help-disk-images```
-help-char-devices	Get help about character *device* specifications. A *device* parameter is required by some emulator options. For example: ```$ emulator -help-char-devices```
-help-sdk-images	Get help about disk images relevant to app developers. It explains where the image files are located for an AVD created with the SDK tools. For example: ```$ emulator -help-sdk-images```
-help-build-images	Get help about disk images relevant to platform developers. For example: ```$ emulator -help-build-images```
-initdata *filepath* -init-data *filepath*	Specify the *initial* version of the data partition. After wiping user data, the emulator copies the contents of the specified file to user data (by default, the userdata-qemu.img file) instead of using the default userdata.img file as the initial version. Specify the filename, and an absolute path or a path relative to the working directory. If you don't specify a path, it places the file in the system directory. For more information, see AVD system directory (#system-filedir). For example: ```$ emulator @Nexus_5X_API_23 -initdata ~/Library/Android/sdk/system-images/android-23/google_apis/x86/userdata-test.img```
-kernel *filepath*	Use a specific emulated kernel. If you don't specify a path, the emulator looks in the system directory. If you don't specify this option, the default is kernel-ranchu. For more information, see AVD system directory (#system-filedir). Use the -show-kernel option to view kernel debug messages. For example: ```$ emulator @Nexus_5X_API_23 -kernel ~/Library/Android/sdk/system-images/android-23/google_apis/x86/kernel-test.img -show-kernel```
-noaudio -no-audio	Disable audio support for this virtual device. Some Linux and Windows computers have faulty audio drivers that cause different symptoms, such as preventing the emulator from starting. In this case, you can use this option to overcome the issue. Alternatively, you can use the QEMU_AUDIO_DRV environment variable to change the audio backend. For example: ```$ emulator @Nexus_5X_API_23 -noaudio```
-nocache -no-cache	Start the emulator without a cache partition. If you don't use this option, the default is a temporary file named cache.img. This option is for platform developers only. For more information, see AVD data directory (#data-filedir). For example: ```$ emulator @Nexus_5X_API_23 -nocache```
-no-snapshot	Inhibit both the automatic load and save operations, causing the emulator to execute a full boot sequence and to lose its state when closed. It overrides the -snapshot option. Note that a snapshot is an experimental feature in Android Studio 2.2. For example: ```$ emulator @Nexus_5X_API_23 -no-snapshot```
-no-snapshot-load	Prevent the emulator from loading the AVD state from snapshot storage. Perform a full boot. Note that a snapshot is an experimental feature in Android Studio 2.2. For example: ```$ emulator @Nexus_5X_API_23 -no-snapshot-load```

`-no-snapshot-save`	Prevent the emulator from saving the AVD state to snapshot storage on exit, meaning that all changes will be lost. Note that a snapshot is an experimental feature in Android Studio 2.2.
	
	For example:
	
	``
	$ emulator @Nexus_5X_API_23 -no-snapshot-save
	``

`-no-snapshot-update-time`	Don't try to correct the AVD clock time immediately on snapshot restore. This option can be useful during testing as it avoids a sudden time jump. Time updates are still sent to the AVD about every 15 seconds, however. Note that a snapshot is an experimental feature in Android Studio 2.2.
	
	For example:
	
	``
	$ emulator @Nexus_5X_API_23 -no-snapshot-update-time
	``

`-no-snapstorage`	Start the emulator without mounting a file to store or load state snapshots, forcing a full boot and disabling state snapshot functionality. This option overrides the `-snapstorage` and `-snapshot` options. Note that a snapshot is an experimental feature in Android Studio 2.2.
	
	For example:
	
	``
	$ emulator @Nexus_5X_API_23 -no-snapstorage
	``

`-no-window`	Disable graphical window display on the emulator. This option is useful when running the emulator on servers that have no display. You'll still be able to access the emulator through adb or the console. For example:
	
	``
	$ emulator @Nexus_5X_API_23 -no-window
	``

`-partition-size` *size*	Specify the system data partition size in MBs. For example:
	
	``
	$ emulator @Nexus_5X_API_23 -partition-size 1024
	``

`-prop` *name*=*value*	Set an Android system property in the emulator when it boots. *name* must be a property name of at most 32 characters, without any spaces in it, and *value* must be a string of at most 92 characters. You can specify several `-prop` options on one command line. This option can be useful for debugging. For example:
	
	``
	$ emulator @Nexus_5X_API_23 -prop status.battery.level_raw=80
	    -prop ro.serialno=123456
	``

`-qemu` *args*	Pass arguments to the QEMU emulator software. Note that QEMU 1 and QEMU 2 can use different arguments. When using this option, make sure it's the last option specified, as all options after it are interpreted as QEMU-specific options. This option is quite advanced and should be used only by developers who are *very* familiar with QEMU *and* Android emulation.

`-qemu -h`	Display `-qemu` help. For example:
	
	``
	$ emulator -qemu -h
	``

`-ramdisk` *filepath*	Specify a ramdisk boot image. Specify the filename, and an absolute path or a path relative to the working directory. If you don't use this option, the default is the `ramdisk.img` file in the system directory. For more information, see AVD system directory (#system-filedir).
	
	For example:
	
	``
	$ emulator @Nexus_5X_API_23 -ramdisk
	    ~/Library/Android/sdk/system-images/android-23/google_apis/x86/ramdisk-test.img
	``

`-report-console` *socket*	Report the console port to a remote third party before starting emulation. It can be useful for an automated testing script. *socket* must use one of these formats: - `tcp:`*port*`[,server][,max=`*seconds*`][,ipv6]` - `unix:`*port*`[,server][,max=`*seconds*`][,ipv6]` For more information, use the `-help-report-console` option as described in Getting detailed help for a specific option (#help-detailed).
`-shell`	Create a root shell console on the current terminal. It differs from the `adb shell` (https://developer.android.com/studio/command-line/adb.html#shellcommands) command in the following ways: - It creates a *root* shell that allows you to modify many parts of the system. - It works even if the adb daemon in the emulated system is broken. - Pressing Ctrl+C (⌘C) stops the emulator, instead of the shell.

For example:

```
$ emulator @Nexus_5X_API_23 -shell
```

-snapshot *name*	Specify the name of a snapshot within a snapshot storage file for automatic start and save operations. Rather than executing a full boot sequence, the emulator can resume execution from an earlier state snapshot, which is usually significantly faster. When you supply this option, the emulator loads the snapshot of that name from the snapshot image and saves it back under the same name on exit. If you don't use this option, the default is a full boot sequence. If the specified snapshot doesn't exist, the emulator performs a full boot sequence instead, and performs a save operation.

See the -snapstorage option for information on specifying a snapshot storage file and the default file.

```
$ emulator @Nexus_5X_API_23 -snapshot snapshot2
```

It's important to remember that in the process of loading a snapshot, all contents of the system, user data, and SD card images are overwritten with the contents they held when the snapshot was made. Unless you save this information in a different snapshot, any changes since then are lost.

You can also create a snapshot from the Emulator Console by using the avd snapshot save *name* command. For more information, see Send Emulator Console Commands to a Virtual Device (https://developer.android.com/studio/run/emulator-console.html).

Note that a snapshot is an experimental feature in Android Studio 2.2.

-snapshot-list	Display a list of available snapshots. It prints a table of snapshots that are stored in the snapshot storage file that the emulator was started with, then exits. If you specify -snapstorage *file* as well, this command prints a table of the snapshots stored in file. Note that a snapshot is an experimental feature in Android Studio 2.2.

For example:

```
$ emulator @Nexus_5X_API_23 -snapshot-list -snapstorage
   ~/.android/avd/Nexus_5X_API_23.avd/snapshots-test.img
```

You can use the ID and TAG column values in the output as arguments for the -snapshot option.

-snapstorage *filepath*	Specify a repository file that contains all state snapshots. All snapshots made during execution will be saved in this file, and only snapshots in this file can be restored during the emulator run. If you don't specify this option, the default is snapshots.img in the data directory. If the specified file doesn't exist, the emulator will start, but without support for saving or loading state snapshots. Note that a snapshot is an experimental feature in Android Studio 2.2.

For example:

```
$ emulator @Nexus_5X_API_23 -snapstorage
   ~/.android/avd/Nexus_5X_API_23.avd/snapshots-test.img
```

-sysdir *dir*	Specify a system directory using an absolute path. For more information, see AVD system directory (#system-filedir). For example:

```
$ emulator @Nexus_5X_API_23 -sysdir
   ~/Library/Android/sdk/system-images/android-23/google_apis/x86/test
```

-system *filepath*	Specify an initial system file. Provide the filename, and an absolute path or a path relative to the working directory. If you don't use this option, the default is the system.img file in the system directory. For more information, see AVD system directory (#system-filedir). For example:

```
$ emulator @Nexus_5X_API_23 -system
   ~/Library/Android/sdk/system-images/android-23/google_apis/x86/system-test.img
```

-use-system-libs	On Linux, use the system libstdc++ instead of the version bundled with the emulator system. Use this option only if the emulator won't start normally, although it doesn't always work. Alternatively, set the ANDROID_EMULATOR_USE_SYSTEM_LIBS (https://developer.android.com/studio/command-line/variables.html#android_emulator_use_system_libs) environment variable to 1.

For example:

```
me-linux$ emulator @Nexus_5X_API_23 -use-system-libs
```

-writable-system	Use this option to have a writable system image during your emulation session. To do so:

1. Start a virtual device with the -writable-system option.

2. Enter the adb remount command from a command terminal to tell the emulator to remount system/ as

	read/write (it's mounted as read-only by default). Note that using this flag will create a temporary copy of the system image that can be very large (several hundred MBs), but will be destroyed when the emulator exits.

Deprecated options

The following command-line options are deprecated:

- `-audio-in`

- `-audio-out`

- `-charmap`

- `-code-profile`

- `-cpu-delay`

- `-dpi-device`

- `-dynamic_skin`

- `-enable-kvm`

- `-gps`

- `-image`

- `-keyset`

- `-help-keys`

- `-help-keyset-file`

- `-nand-limits`

- `-noskin`

- `-no-skin`

- `-onion`

- `-onion-alpha`

- `-onion-rotation`

- `-radio`

- `-ranchu`

- `-raw-keys`

- `-scale`

- `-shared-net-id`

- `-shell-serial`

- `-skin`

- `-skindir`

- `-trace`

- `-useaudio`

Getting help about command-line options

This section describes how to get help about the command-line options. The following section provides more in-depth information about the commonly used emulator command-line options that are available when you start the emulator.

Listing all emulator options

To print a list of all emulator options, including a short description, enter this command:

```
emulator -help
```

Getting detailed help for a specific option

To print help for a specific startup option, enter this command:

```
emulator -help-option
```

For example:

```
emulator -help-netspeed
```

This help is more detailed than the description provided by the `-help` option.

Getting detailed help for all options

To get detailed help for all emulator options, enter this command:

```
emulator -help-all
```

Listing emulator environment variables

To get a list of emulator environment variables, enter this command:

```
emulator -help-environment
```

You can set environment variables in the terminal window before launching a virtual device, or you could set it through your user settings in the operating system; for example, in your `.bashrc` file on Linux.

Listing debug tags

To print a list of tags for the `-debug` options, enter this command:

```
emulator -help-debug-tags
```

The `-debug` options let you enable or disable debug messages from specific emulator components, as specified by the tags.

Follow @AndroidDev
on Twitter

Follow Android Developers
on Google+

Check out Android Developers
on YouTube