Android 调试桥

Android 调试桥 (adb) 是一个通用命令行工具，其允许您与模拟器实例或连接的 Android 设备进行通信。它可为各种设备操作提供便利，如安装和调试应用，并提供对 Unix shell（可用来在模拟器或连接的设备上运行各种命令）的访问。该工具作为一个客户端-服务器程序，包括三个组件：

- **客户端**，该组件发送命令。客户端在开发计算机上运行。您可以通过发出 adb 命令从命令行终端调用客户端。
- **后台程序**，该组件在设备上运行命令。后台程序在每个模拟器或设备实例上作为后台进程运行。
- **服务器**，该组件管理客户端和后台程序之间的通信。服务器在开发计算机上作为后台进程运行。

您可以在 `android_sdk/platform-tools/` 中找到 `adb` 工具。

adb 的工作方式

启动一个 adb 客户端时，此客户端首先检查是否有已运行的 adb 服务器进程。如果没有，它将启动服务器进程。当服务器启动时，它与本地 TCP 端口 5037 绑定，并侦听从 adb 客户端发送的命令—所有 adb 客户端均使用端口 5037 与 adb 服务器通信。

然后，服务器设置与所有运行的模拟器/设备实例的连接。它通过扫描 5555 到 5585 之间（模拟器/设备使用的范围）的奇数号端口查找模拟器/设备实例。服务器一旦发现 adb 后台程序，它将设置与该端口的连接。请注意，每个模拟器/设备实例将获取一对按顺序排列的端口 — 用于控制台连接的偶数号端口和用于 adb 连接的奇数号端口。例如：

模拟器 1，控制台：5554　　模拟器 1，adb：5555
模拟器 2，控制台：5556　　模拟器 2，adb：5557
以此类推...

如上所示，在端口 5555 与 adb 连接的模拟器实例与侦听端口 5554 的控制台的实例相同。

当服务器已设置与所有模拟器实例的连接后，您可以使用 adb 命令访问这些实例。由于服务器管理与模拟器/设备实例的连接，并处理来自多个 adb 客户端的命令，因此，您可以从任意客户端（或从某个脚本）控制任意模拟器/设备实例。

在您的设备上启用 adb 调试

要在通过 USB 连接的设备上使用 adb，您必须在设备系统设置中启用 **USB debugging**（位于 **Developer options** 下）。

在运行 Android 4.2 及更高版本的设备上，Developer options 屏幕默认情况下处于隐藏状态。如需将其显示出来，请转到 **Settings > About phone** 并点按 **Build number** 七次。返回上一屏幕，在底部可以找到 **Developer options**。

在某些设备上，Developer options 屏幕所在的位置或命名方式可能有所不同。

现在，您可以将设备与 USB 连接。可以从 `android_sdk/platform-tools/` 目录执行 `adb devices` 来验证设备是否连接。如果已连接，您将看到设备名称以“设备”形式列示。

注：当您连接运行 Android 4.2.2 或更高版本的设备时，系统将显示一个对话框，询问您是否接受允许在这台计算机上调试的 RSA 密钥。这种安全机制可以保护用户设备，因为它可以确保只有在您能够解锁设备并确认对话框的情况下才能执行 USB 调试和其他 ADB 命令。

如需了解有关通过 USB 连接到设备的详细信息，请参阅在硬件设备上运行应用。

通过 WLAN 连接到设备

一般情况下，通过 USB 使用 adb。不过，也可以按照下面的说明通过 WLAN 使用它。

1. 将 Android 设备和 adb 主计算机连接到这两者都可以访问的常用 WLAN 网络。请注意，并非所有访问点均适用；您可能需要使用已正确配置防火墙的访问点以支持 adb 的访问点。

注： 如果您尝试连接到 Android Wear 设备，则通过关闭与其连接的手机的蓝牙强制将它连接到 WLAN。

2. 使用 USB 电缆将设备连接到主计算机。
3. 设置目标设备以侦听端口 5555 上的 TCP/IP 连接。

```
$ adb tcpip 5555
```

4. 从目标设备断开 USB 电缆连接。
5. 查找 Android 设备的 IP 地址。例如，在 Nexus 设备上，您可以通过访问 **Settings** > **About tablet**（或 **About phone**）> **Status** > **IP address** 查找 IP 地址。或者，在 Android Wear 设备上，您可以通过访问 **Settings** > **Wi-Fi Settings** > **Advanced** > **IP address** 查找 IP 地址。
6. 连接至设备，通过 IP 地址识别此设备。 `$ adb connect device_ip_address`
7. 请确认您的主计算机已连接至目标设备：

```
$ adb devices
List of devices attached
device_ip_address:5555  device
```

现在，您可以开始操作了！ 如果 adb 连接丢失：

1. 请确保您的主机仍与您的 Android 设备连接到同一个 WLAN 网络。
2. 通过再次执行 `adb connect` 步骤重新连接。
3. 如果无法连接，则重置 adb 主机： `adb kill-server` 然后，从头开始操作。

查询设备

在发出 adb 命令之前，知道哪些模拟器/设备实例已连接到 adb 服务器会很有帮助。您可以使用 `devices` 命令生成已连接的模拟器/设备的列表： **`adb devices`** 在响应时，adb 针对每个实例输出此状态信息：

- 序列号 — 一个由 adb 创建的字符串，用于通过其控制台端口号唯一标识模拟器/设备实例。序列号的格式为 *`type-console-port`*。下面是一个序列号示例：`emulator-5554`
- 状态 — 实例的连接状态可为下列状态之一：
- `offline` — 实例未连接到 adb 或不响应。
- `device` — 实例现在已连接到 adb 服务器。请注意，此状态并不表示 Android 系统已完全启动且可以运行，因为在此实例连接到 adb 时系统仍在启动。不过，在启动后，这将是模拟器/设备实例的正常运行状态。
- `no device` — 未连接模拟器/设备。

输出的格式类似如下：

```
List of devices attached

serial_number state
```

以下示例向您展示了 `devices` 命令及其输出：**adb devices**

List of devices attached
emulator-5556 device
emulator-5558 device

将命令发送至特定设备

如果多个模拟器/设备实例正在运行，在发出 adb 命令时您必须指定一个目标实例。为此，请在命令中使用 `-s` 选项。以下是 `-s` 选项的用法：`adb -s serial_number command`

如上所示，您使用由 adb 分配的序列号为命令指定目标实例。您可使用 `devices` 命令获取正在运行的模拟器/设备实例的序列号。例如：`adb -s emulator-5556 install helloWorld.apk`

注意，如果在多个设备可用时您未指定目标模拟器/设备实例就发出命令，那么 adb 将生成一个错误。

如果您有多个设备可用（硬件或模拟设备），但只有一个设备是模拟器，则使用 `-e` 选项将命令发送至该模拟器。同样，如果有多个设备，但只连接了一个硬件设备，则使用 `-d` 选项将命令发送至该硬件设备。

安装应用

您可以使用 adb 从开发计算机复制应用，并将其安装到模拟器/设备实例上。为此，请使用 `install` 命令。使用此命令，您必须指定您要安装的 APK 文件的路径：`adb install path_to_apk`

如需有关如何创建可在模拟器/设备实例上安装的 APK 文件的详细信息，请参阅构建和运行您的应用。

请注意，如果使用 Android Studio，则无需直接使用 adb（或 aapt）在模拟器/设备上安装您的应用。而是由 Android Studio 为您处理应用的打包和安装。

设置端口转发

您可以使用 `forward` 命令设置任意端口转发 — 将对特定主机端口的请求转发到模拟器/设备实例上的其他端口。下面向您介绍如何设置主机端口 6100 到模拟器/设备端口 7100 的转发：`adb forward tcp:6100 tcp:7100`

您也可以使用 adb 设置传输到指定的抽象 UNIX 网域套接字的转发，如下所示：`adb forward tcp:6100 local:logd`

将文件复制到设备/从设备复制文件

您可以使用 adb 命令 `pull` 和 `push` 将文件复制到模拟器/设备实例或从其中复制文件。与 `install` 命令不同（其仅将 APK 文件复制到特定位置），`pull` 和 `push` 命令允许您将任意目录和文件复制到模拟器/设备实例中的任意位置。

要*从*模拟器或设备复制文件或目录（及其子目录），请使用 `adb pull remote local`

要将文件文件或目录（及其子目录）复制*到*模拟器或设备，请使用 `adb push local remote`

在上述命令中，`local` 和 `remote` 指的是开发计算机（本地）和模拟器/设备实例（远程）上目标文件/目录的路径。例如：

```
adb push foo.txt /sdcard/foo.txt
```

停止 adb 服务器

在某些情况下，您可能需要终止 adb 服务器进程，然后重启它以解决问题（例如，如果 adb 不响应命令）。

要停止 adb 服务器，请使用 `adb kill-server` 命令。然后，您可以通过发出任意其他 adb 命令重启服务器。

adb 命令参考

您可以在开发计算机上从命令行发出 adb 命令，或通过脚本发出。用法如下：

```
adb [-d|-e|-s serial_number] command
```

如果只有一个模拟器在运行或只连接了一个设备，则默认情况下将 adb 命令发送至该设备。如果有多个模拟器在运行和/或连接了多个设备，您需要使用 `-d`、`-e` 或 `-s` 选项指定应向其发送命令的目标设备。

下表列出了所有支持的 adb 命令并解释其含义和用法。

表 1. 可用的 adb 命令

类别	命令	说明	注释
目标设备	`-d`	将 adb 命令发送至唯一连接的 USB 设备。	如果连接了多个 USB 设备，将返回错误。
	`-e`	将 adb 命令发送至唯一运行的模拟器实例。	如果有多个模拟器实例在运行，将返回错误。
	`-s serial_number`	将 adb 命令发送至以其 adb 分配的序列号命名的特定模拟器/设备实例（如“emulator-5556”）。	请参阅将命令发送至特定模拟器/设备实例。
常规	`devices`	输出所有连接的模拟器/设备实例的列表。	如需了解详细信息，请参阅查询模拟器/设备实例。
	`help`	输出支持的 adb 命令的列表。	
	`version`	输出 adb 版本号。	
调试	`logcat [option] [filter-specs]`	将日志数据输出到屏幕。	
	`bugreport`	将 `dumpsys`、`dumpstate` 和 `logcat` 数据输出到屏幕，以用于报告错误。	

	`jdwp`	输出给定设备上可用的 JDWP 进程的列表。	您可以使用 `forward jdwp:pid` 端口转发规范以连接到特定的 JDWP 进程。例如：
			`adb forward tcp:8000 jdwp:472`
			`jdb -attach localhost:8000`

数据	`install path_to_apk`	将 Android 应用（使用 APK 文件的完整路径表示）推送到模拟器/设备。	
	`pull remote local`	从模拟器/设备实例将指定文件复制到开发计算机。	

	`push local remote`	开发计算机将指定文件复制到模拟器/设备实例	

端口和网络连接	`forward local remote`	将来自指定本地端口的套接字连接转发到模拟器/设备实例上的指定远程端口。	端口规范可使用以下架构：
			`tcp:port_number`
			`local:unix_domain_socket_name`
			`dev:character_device_name`
			`jdwp:pid`

	`ppp tty [parm]...`	通过 USB 运行 PPP。*tty* — 用于 PPP 流的 tty。例 `dev:/dev/omap_csmi_ttyl`。 `[parm]`. — 零个或多个 PPP/PPPD 选项，如 `defaultroute`、`local`、`notty` 等。 请注意，不得自动启动 PPP 连接。	
脚本	`get-serialno`	输出 adb 实例序列号字符串。	如需了解详细信息，请参阅查询模拟器/设备实例。
	`get-state`	输出模拟器/设备实例的 adb 状态。	
	`wait-for-device`	阻止执行，直至设备处于在线状态，即直至此实例状态为 `device`。	您可以将此命令附加到其他 adb 命令，在此情况下，adb 在发出其他命令前将处于等待状态，直至模拟器/设备实例已连接。下面是一个示例： `adb wait-for-device shell getprop` 请注意，此命令*不会*使 adb 等待整个系统已完全启动。因此，您不应将其追加到需要系统完全启动的其他命令。例如，`install` 需要使用 Android 软件包管理器，其仅在系统完全启动后才可用。如下命令 `adb wait-for-device install app.apk` 在模拟器或设备实例连接到 adb 服务器时立即发出 `install` 命令，但 Android 系统还未完全启动，因此，它将引发错误。
服务器	`start-server`	检查 adb 服务器进程是否在运行，如果未运行则启动它。	
	`kill-server`	终止 adb 服务器进程。	
Shell	`shell`	在目标模拟器/设备实例中启动远程 shell。	如需了解详细信息，请参阅发出 shell 命令。
	`shell shell_command`	在目标模拟器/设备实例中发出 shell 命令，然后退出远程 shell。	

发出 shell 命令

您可以使用 `shell` 命令通过 adb 发出设备命令，可以进入或不进入模拟器/设备实例上的 adb 远程 shell。要在不进入远程 shell 的情况下发出一个命令，请使用如下 `shell` 命令： `adb [-d|-e|-s serial_number] shell shell_command`

或者，使用如下命令进入模拟器/设备实例上的远程 shell： `adb [-d|-e|-s serial_number] shell`

当您准备退出远程 shell 时，按 Control + D 或输入 `exit`。

shell 命令二进制文件存储在模拟器或设备的文件系统中，其路径为 `/system/bin/`。

调用 Activity Manager (am)

在 adb shell 中，您可以使用 Activity Manager (am) 工具发出命令以执行各种系统操作，如启动 Activity、强行停止进程、广播 intent、修改设备屏幕属性及其他操作。在 shell 中，此语法为： `am command`

您也可以直接从 adb 发出 Activity Manager 命令，无需进入远程 shell。例如：

```
adb shell am start -a android.intent.action.VIEW
```

表 2. 可用的 Activity Manager 命令

命令	说明
`start [options] intent`	启动 *intent* 指定的 Activity。请参阅 intent 参数的规范。选项包括： `-D`：启用调试。 `-W`：等待启动完成。 `--start-profiler file`：启动分析器并将结果发送到 *file*。 `-P file`：类似于 `--start-profiler`，但当应用进入空闲状态时分析停止。 `-R count`：重复 Activity 启动 *count* 次数。在每次重复前，将完成顶部 Activity。 `-S`：启动 Activity 前强行停止目标应用。 `--opengl-trace`：启用 OpenGL 函数的跟踪。 `--user user_id | current`：指定要作为哪个用户运行；如果未指定，则作为当前用户运行。
`startservice [options] intent`	启动 *intent* 指定的 Service。请参阅 intent 参数的规范。选项包括： `--user user_id | current`：指定要作为哪个用户运行；如果未指定，则作为当前用户运行。
`force-stop package`	强行停止与 *package*（应用的包名称）关联的所有应用。
`kill [options] package`	终止与 *package*（应用的包名称）关联的所有进程。此命令仅终止可安全终止且不会影响用户体验的进程。选项包括： `--user user_id | all | current`：指定将终止其进程的用户；如果未指定，则终止所有用户的进程。
`kill-all`	终止所有后台进程。
`broadcast [options] intent`	发出广播 intent。请参阅 intent 参数的规范。选项包括： `[--user user_id | all | current]`：指定要发送到的用户；如果未指定，则发送到所有用户。
`instrument [options] component`	使用 Instrumentation 实例启动监控。通常，目标 *component* 是表单 *test_package*/*runner_class*。 选项包括： `-r`：输出原始结果（否则对 *report_key_streamresult* 进行解码）。与 `[-e perf true]` 结合使用以生成性能测量的原始输出。 `-e name value`：将参数 *name* 设为 *value*。对于测试运行器，通用表单为 `-e testrunner_flag value[,value...]`。 `-p file`：将分析数据写入 *file*。 `-w`：先等待仪器完成，然后再返回。测试运行器需要使用此选项。 `--no-window-animation`：运行时关闭窗口动画。 `--user user_id | current`：指定仪器在哪个用户中运行；如果未指定，则在当前用户中运行。

`profile start` ***process file***	启动***process***的分析器，将结果写入***file***。
`profile stop` ***process***	停止***process***的分析器。
`dumpheap [`***options***`]` ***process file***	转储***process***的堆，写入***file***。选项包括： `--user [`***user_id***`|current]`：提供进程名称时，指定要转储的进程用户；如未指定，则使用当前用户。 `-n`：转储原生堆，而非托管堆。
`set-debug-app [`***options***`]` ***package***	将应用***package***设为调试。选项包括：`-w`：应用启动时等待调试程序。 `--persistent`：保留此值。
`clear-debug-app`	使用 `set-debug-app` 清除以前针对调试用途设置的软件包。
`monitor [`***options***`]`	启动对崩溃或 ANR 的监控。选项包括： `--gdb`：在崩溃/ANR 时在给定端口上启动 gdbserv。
`screen-compat {on|off}` ***package***	控制***package***的屏幕兼容性模式。
`display-size [reset|`***width***`x`***height***`]`	替换模拟器/设备显示尺寸。此命令对于在不同尺寸的屏幕上测试您的应用非常有用，它支持使用大屏设备模仿小屏幕分辨率（反之亦然）。 示例： `am display-size 1280x800`
`display-density` ***dpi***	替换模拟器/设备显示密度。此命令对于在不同密度的屏幕上测试您的应用非常有用，它支持使用低密度屏幕在高密度环境环境上进行测试（反之亦然）。 示例： `am display-density 480`
`to-uri` ***intent***	将给定的 intent 规范以 URI 的形式输出。请参阅 intent 参数的规范。
`to-intent-uri` ***intent***	将给定的 intent 规范以 `intent:`URI 的形式输出。请参阅 intent 参数的规范。

intent 参数的规范

对于采用 intent 参数的 Activity Manager 命令，您可以使用以下选项指定 intent：

-a action 指定 intent 操作，如 “android.intent.action.VIEW” 。此指定只能声明一次。
-d data_uri 指定 intent 数据 URI，如 “content://contacts/people/1” 。此指定只能声明一次。
-t mime_type 指定 intent MIME 类型，如 “image/png” 。此指定只能声明一次。
-c category 指定 intent 类别，如 “android.intent.category.APP_CONTACTS” 。
-n component 指定带有软件包名称前缀的组件名称以创建显式 intent，如 “com.example.app/.ExampleActivity” 。
-f flags 将标志添加到 setFlags() 支持的 intent。
--esn extra_key 添加一个 null extra。URI intent 不支持此选项。
-e|--es extra_key extra_string_value 添加字符串数据作为键值对。
--ez extra_key extra_boolean_value 添加布尔型数据作为键值对。
--ei extra_key extra_int_value 添加整数型数据作为键值对。
--el extra_key extra_long_value 添加长整型数据作为键值对。
--ef extra_key extra_float_value 添加浮点型数据作为键值对。
--eu extra_key extra_uri_value 添加 URI 数据作为键值对。
--ecn extra_key extra_component_name_value 添加组件名称，将其作为 ComponentName 对象进行转换和传递。
--eia extra_key extra_int_value[,extra_int_value...] 添加整数数组。
--ela extra_key extra_long_value[,extra_long_value...] 添加长整型数组。
--efa extra_key extra_float_value[,extra_float_value...] 添加浮点型数组。
--grant-read-uri-permission 包含标志 FLAG_GRANT_READ_URI_PERMISSION。
--grant-write-uri-permission 包含标志 FLAG_GRANT_WRITE_URI_PERMISSION。
--debug-log-resolution 包含标志 FLAG_DEBUG_LOG_RESOLUTION。
--exclude-stopped-packages 包含标志 FLAG_EXCLUDE_STOPPED_PACKAGES。
--include-stopped-packages 包含标志 FLAG_INCLUDE_STOPPED_PACKAGES。
--activity-brought-to-front 包含标志 FLAG_ACTIVITY_BROUGHT_TO_FRONT。

--activity-clear-top 包含标志 FLAG_ACTIVITY_CLEAR_TOP。
--activity-clear-when-task-reset 包含标志 FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET。
--activity-exclude-from-recents 包含标志 FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS。
--activity-launched-from-history 包含标志 FLAG_ACTIVITY_LAUNCHED_FROM_HISTORY。
--activity-multiple-task 包含标志 FLAG_ACTIVITY_MULTIPLE_TASK。
--activity-no-animation 包含标志 FLAG_ACTIVITY_NO_ANIMATION。
--activity-no-history 包含标志 FLAG_ACTIVITY_NO_HISTORY。
--activity-no-user-action 包含标志 FLAG_ACTIVITY_NO_USER_ACTION。
--activity-previous-is-top 包含标志 FLAG_ACTIVITY_PREVIOUS_IS_TOP。
--activity-reorder-to-front 包含标志 FLAG_ACTIVITY_REORDER_TO_FRONT。
--activity-reset-task-if-needed 包含标志 FLAG_ACTIVITY_RESET_TASK_IF_NEEDED。
--activity-single-top 包含标志 FLAG_ACTIVITY_SINGLE_TOP。
--activity-clear-task 包含标志 FLAG_ACTIVITY_CLEAR_TASK。
--activity-task-on-home 包含标志 FLAG_ACTIVITY_TASK_ON_HOME。
--receiver-registered-only 包含标志 FLAG_RECEIVER_REGISTERED_ONLY。
--receiver-replace-pending 包含标志 FLAG_RECEIVER_REPLACE_PENDING。
--selector 需要使用 -d 和 -t 选项以设置 intent 数据和类型。

URI component package

如果不受上述某一选项的限制，您可以直接指定 URI、软件包名称和组件名称。当参数不受限制时，如果参数包含一个“:”（冒号），则此工具假定参数是一个 URI；如果参数包含一个“/”（正斜杠），则此工具假定参数是一个组件名称；否则，此工具假定参数是一个软件包名称。

调用软件包管理器 (pm)

在 adb shell 中，您可以使用软件包管理器 (pm) 工具发出命令，以对设备上安装的应用软件包进行操作和查询。在 shell 中，此语法为： `pm command`

您也可以直接从 adb 发出软件包管理器命令，无需进入远程 shell。例如： `adb shell pm uninstall com.example.MyApp`

表 3. 可用的软件包管理器命令。

命令	说明
`list packages [options] filter`	输出所有软件包，或者，仅输出包名称包含 *filter* 中的文本的软件包。选项： `-f`：查看它们的关联文件。 `-d`：进行过滤以仅显示已停用的软件包。 `-e`：进行过滤以仅显示已启用的软件包。 `-s`：进行过滤以仅显示系统软件包。 `-3`：进行过滤以仅显示第三方软件包。 `-i`：查看软件包的安装程序。 `-u`：也包括卸载的软件包。 `--user user_id`：要查询的用户空间。
`list permission-groups`	输出所有已知的权限组。
`list permissions [options] group`	输出所有已知权限，或者，仅输出 *group* 中的权限。选项： `-g`：按组加以组织。 `-f`：输出所有信息。 `-s`：简短摘要。 `-d`：仅列出危险权限。 `-u`：仅列出用户将看到的权限。
`list instrumentation [options]`	列出所有测试软件包。选项： `-f`：列出用于测试软件包的 APK 文件。 *target_package*：列出仅用于此应用的测试软件包。

list features	输出系统的所有功能。
list libraries	输出当前设备支持的所有库。
list users	输出系统上的所有用户。
path *package*	输出给定*package*的 APK 的路径。
install [*options*] *path*	将软件包（通过*path*指定）安装到系统。 选项： -l：安装具有转发锁定功能的软件包。 -r：重新安装现有应用，保留其数据。 -t：允许安装测试 APK。 -i *installer_package_name* 指定安装程序软件包名称。 -s：在共享的大容量存储（如 sdcard）上安装软件包。 -f：在内部系统内存上安装软件包。 -d：允许版本代码降级。 -g：授予应用清单中列出的所有权限。
uninstall [*options*] *package*	从系统中移除软件包。 选项：-k：移除软件包后保留数据和缓存目录。
clear *package*	删除与软件包关联的所有数据。
enable *package_or_component*	启用给定软件包或组件（作为“package/class”写入）。
disable *package_or_component*	停用给定软件包或组件（作为“package/class”写入）。
disable-user [*options*] *package_or_component*	选项：--user *user_id*：要停用的用户。
grant *package_name permission*	向应用授予权限。在运行 Android 6.0（API 级别 23）及更高版本的设备上，可以是应用清单中声明的任何权限。在运行 Android 5.1（API 级别 22）和更低版本的设备上，必须是应用定义的可选权限。
revoke *package_name permission*	从应用中撤销权限。在运行 Android 6.0（API 级别 23）及更高版本的设备上，可以是应用清单中声明的任何权限。在运行 Android 5.1（API 级别 22）和更低版本的设备上，必须是应用定义的可选权限。
set-install-location *location*	更改默认安装位置。位置值： 0：自动—让系统决定最佳位置。 1：内部—安装在内部设备存储上。 2：外部—安装在外部介质上。 注：此命令仅用于调试目的；使用此命令会导致应用中断和其他意外行为。
get-install-location	返回当前安装位置。返回值： 0 [auto]：让系统决定最佳位置。 1 [internal]：安装在内部设备存储上 2 [external]：安装在外部介质上
set-permission-enforced *permission* [true|false]	指定是否应强制执行给定的权限。
trim-caches *desired_free_space*	减少缓存文件以达到给定的可用空间。
create-user *user_name*	使用给定的*user_name*创建新用户，输出新用户的标识符。
remove-user *user_id*	移除具有给定的*user_id*的用户，删除与该用户关联的所有数据。
get-max-users	输出设备支持的最大用户数。

进行屏幕截图

`screencap` 命令是一个用于对设备显示屏进行屏幕截图的 shell 实用程序。在 shell 中，此语法为： `screencap filename`

要从命令行使用 `screencap`，请输入以下命令： `$ adb shell screencap /sdcard/screen.png`

以下屏幕截图会话示例向您展示使用 adb shell 捕获屏幕截图，并使用 `pull` 命令从设备下载此文件：

```
$ adb shell
shell@ $ screencap /sdcard/screen.png
shell@ $ exit
$ adb pull /sdcard/screen.png
```

录制视频

`screenrecord` 命令是一个用于录制设备（运行 Android 4.4（API 级别 19）及更高版本）显示屏的 shell 实用程序。此实用程序将屏幕 Activity 录制到 MPEG-4 文件。

注：音频不与视频文件一起录制。

开发者可以使用此文件创建宣传视频或培训视频。在 shell 中，此语法为： `screenrecord [options] filename`

要从命令行使用 `screenrecord`，请输入以下命令： `$ adb shell screenrecord /sdcard/demo.mp4`

按 Control + C 停止屏幕录制，否则，到三分钟或 `--time-limit` 设置的时间限制时，录制将自动停止。

要开始录制设备屏幕，请运行 `screenrecord` 命令以录制视频。然后，运行 `pull` 命令从设备将此视频下载到主计算机。下面是一个录制会话示例：

```
$ adb shell
shell@ $ screenrecord --verbose /sdcard/demo.mp4
(press Control + C to stop)
shell@ $ exit
$ adb pull /sdcard/demo.mp4
```

`screenrecord` 实用程序可以任何支持的分辨率和所需的比特率进行录制，同时保留设备显示屏的纵横比。默认情况下，此实用程序以原生显示分辨率和屏幕方向进行录制，最大时长为三分钟。

下面是 `screenrecord` 实用程序的一些已知限制，您在使用时应注意：

- 某些设备可能无法以它们的原生显示分辨率进行录制。如果在录制屏幕时出现问题，请尝试使用较低的屏幕分辨率。
- 不支持在录制时旋转屏幕。如果在录制期间屏幕旋转了，则部分屏幕的录制将被切断。

表 4. `screenrecord` 选项

选项	说明
`--help`	显示命令语法和选项
`--size` ***`width`***`x`***`height`***	设置视频大小：`1280x720`。默认值是设备的原生显示分辨率（如果支持），如果不支持，则使用 1280x720。为实现最佳结果，请使用设备的 Advanced Video Coding (AVC) 编码器支持的大小。
`--bit-rate` ***`rate`***	设置视频的视频比特率（以兆比特每秒为单位）。默认值为 4Mbps。您可以增加比特率以提升视频质量，但这么做会导致影片文件变得更大。以下示例将录制比特率设为 6Mbps： `screenrecord --bit-rate 6000000 /sdcard/demo.mp4`
`--time-limit` ***`time`***	设置最大录制时长（以秒为单位）。默认值和最大值均为 180（3 分钟）。
`--rotate`	将输出旋转 90 度。此功能是实验性的。
`--verbose`	显示命令行屏幕上的日志信息。如果您不设置此选项，则运行时此实用程序不会显示任何信息。

读取应用的 ART 配置文件

从 Android 7.0（API 级别 24）开始，Android Runtime (ART) 会收集已安装应用的执行配置文件，其可用于优化应用性能。您可能想要检查收集的配置文件，以了解在应用启动期间，系统决定频繁执行哪些方法和使用哪些类。

要生成配置文件信息的文本表单，请使用以下命令： `$ adb shell cmd package dump-profiles` ***`package`***

要检索生成的文件，请使用： `$ adb pull /data/misc/profman/`***`package`***`.txt`

其他 shell 命令

如需所有可用 shell 程序的列表，请使用以下命令： **`adb shell ls /system/bin`** 大多数命令都提供帮助说明。

表 5. 其他一些 adb shell 命令

Shell 命令	说明	注释
`dumpsys`	将系统数据转储到屏幕。	Dalvik Debug Monitor Server (DDMS) 工具提供了一个集成调试环境，让您用起来更方便。
`dumpstate`	将状态转储到文件。	
`logcat [option]... [filter-spec]...`	启用系统和应用日志记录，并将输出传输到屏幕。	
`dmesg`	将内核调试消息输出到屏幕。	
`start`	启动（重启）模拟器/设备实例。	
`stop`	停止执行模拟器/设备实例。	

上次更新日期：四月 25, 2018