# Мост отладки Android (adb)

Android Debug Bridge ( `adb`) — это универсальный инструмент командной строки, позволяющий взаимодействовать с устройством. Команда `adb`облегчает различные действия с устройством, такие как установка и отладка приложений. `adb`предоставляет доступ к оболочке Unix, которую можно использовать для запуска различных команд на устройстве. Это клиент-серверная программа, включающая в себя три компонента:

- **Клиент** , который отправляет команды. Клиент работает на вашей машине разработки. Вы можете вызвать клиента из терминала командной строки, введя команду `adb`.
- **Демон (adbd)** , который запускает команды на устройстве. Демон работает как фоновый процесс на каждом устройстве.
- **Сервер** , который управляет связью между клиентом и демоном. Сервер работает как фоновый процесс на вашей машине разработки.

`adb`включен в пакет инструментов платформы Android SDK. Загрузите этот пакет с помощью SDK Manager (/studio/intro/update#sdk-manager) , который установит его по адресу `android_sdk/platform-tools/`. Если вам нужен автономный пакет инструментов платформы Android SDK, загрузите его здесь (/studio/releases/platform-tools) .

Информацию о подключении устройства для использования с ним `adb`, в том числе об использовании Помощника по подключению для устранения распространенных проблем, см. в разделе Запуск приложений на аппаратном устройстве (/studio/run/device) .

## Как работает адб

Когда вы запускаете `adb`клиент, он сначала проверяет, запущен ли `adb`уже серверный процесс. Если его нет, он запускает серверный процесс. Когда сервер запускается, он подключается к локальному TCP-порту 5037 и прослушивает команды, отправленные от `adb`клиентов.

**Примечание.** Все `adb`клиенты используют порт 5037 для связи с `adb`сервером.

Затем сервер устанавливает соединения со всеми работающими устройствами. Он находит эмуляторы путем сканирования портов с нечетными номерами в диапазоне от 5555 до 5585, который используется первыми 16 эмуляторами. Когда сервер находит `adb` демон (adbd), он устанавливает соединение с этим портом.

Каждый эмулятор использует пару последовательных портов — порт с четным номером для консольных подключений и порт с нечетным номером для `adb`подключений. Например:

Эмулятор 1, консоль: 5554
Эмулятор 1, `adb`: 5555
Эмулятор 2, консоль: 5556
Эмулятор 2, `adb`: 5557
и так далее.

Как показано, эмулятор, подключенный к `adb`порту 5555, аналогичен эмулятору, консоль которого прослушивает порт 5554.

После того как сервер настроит соединения со всеми устройствами, вы сможете использовать `adb`команды для доступа к этим устройствам. Поскольку сервер управляет подключениями к устройствам и обрабатывает команды от нескольких `adb`клиентов, вы можете управлять любым устройством с любого клиента или из сценария.

## Включите отладку adb на вашем устройстве

Чтобы использовать adb с устройством, подключенным через USB, необходимо включить **отладку по USB** в настройках системы устройства в разделе **«Параметры разработчика»** . В Android 4.2 (уровень API 17) и более поздних версиях экран **параметров разработчика** по умолчанию скрыт. Чтобы сделать его видимым, включите параметры разработчика. (/studio/debug/dev-options#enable)

Теперь вы можете подключить свое устройство через USB. Вы можете убедиться, что ваше устройство подключено, выполнив команду `adb devices`из `android_sdk/platform-tools/`каталога. Если оно подключено, вы увидите имя устройства в списке «устройство».

**Примечание.** При подключении устройства под управлением Android 4.2.2 (уровень API 17) или выше система отображает диалоговое окно с вопросом, принять ли ключ RSA, позволяющий отладку через этот компьютер. Этот механизм безопасности защищает пользовательские

устройства, поскольку он гарантирует, что отладка USB и другие команды adb не могут быть выполнены, если вы не сможете разблокировать устройство и подтвердить диалог.

Дополнительные сведения о подключении к устройству через USB см. в статье Запуск приложений на аппаратном устройстве (/studio/run/device) .

## Подключитесь к устройству через Wi-Fi

**Примечание.** Приведенные ниже инструкции не применимы к устройствам Wear под управлением Android 11 (уровень API 30). Дополнительную информацию см. в руководстве по отладке приложения Wear OS . (/training/wearables/get-started/debugging#wifi-debugging)

Android 11 (уровень API 30) и более поздние версии поддерживают развертывание и отладку приложения по беспроводной сети с рабочей станции с помощью Android Debug Bridge (adb). Например, вы можете развернуть отлаживаемое приложение на нескольких удаленных устройствах без необходимости физического подключения устройства через USB. Это избавляет от необходимости решать распространенные проблемы с подключением USB, например установку драйверов.

Прежде чем начать использовать беспроводную отладку, выполните следующие действия:

- Убедитесь, что ваша рабочая станция и устройство подключены к одной беспроводной сети.
- Убедитесь, что ваше устройство работает под управлением Android 11 (уровень API 30) или более поздней версии для телефона или Android 13 (уровень API 33) или более поздней версии для телевизора и WearOS. Дополнительную информацию см. в разделе Проверка и обновление версии Android (https://support.google.com/android/answer/7680439) .
- Если вы используете IDE, убедитесь, что у вас установлена последняя версия Android Studio. Вы можете скачать это здесь (/studio) .
- На своей рабочей станции обновите SDK Platform Tools (/studio/releases/platform-tools) до последней версии .

Чтобы использовать беспроводную отладку, необходимо выполнить сопряжение устройства с рабочей станцией с помощью QR-кода или кода сопряжения. Ваша рабочая станция и устройство должны быть подключены к одной беспроводной сети. Чтобы подключиться к вашему устройству, выполните следующие действия:

1. Включите параметры разработчика (/studio/debug/dev-options#enable) на вашем устройстве.
2. Откройте Android Studio и выберите **«Подключить устройства с помощью Wi-Fi»** в меню настроек запуска.

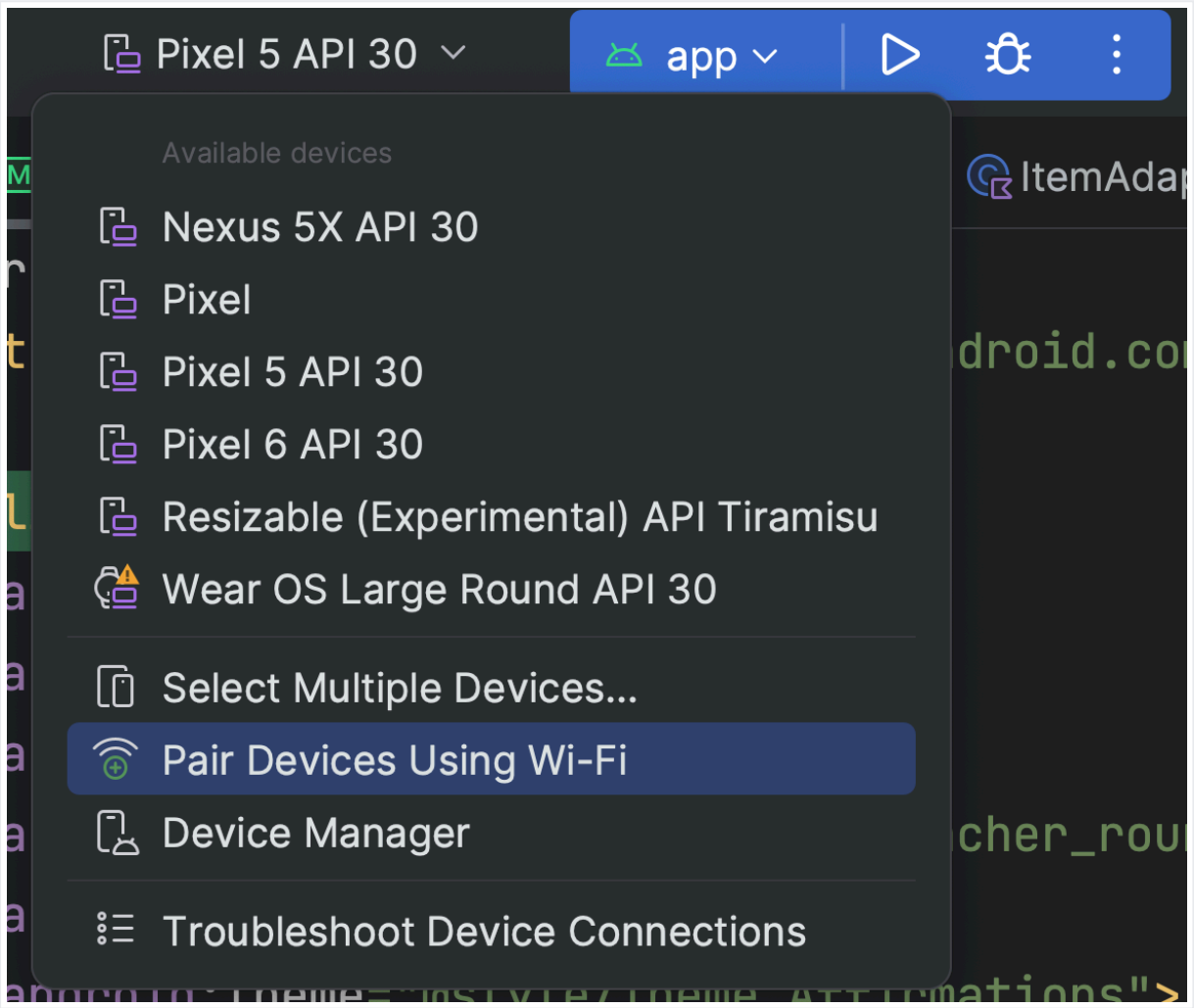


**Рисунок 1.** Меню запуска настроек.

Появится окно «Сопряжение устройств через Wi-Fi», как показано на рисунке 2 .

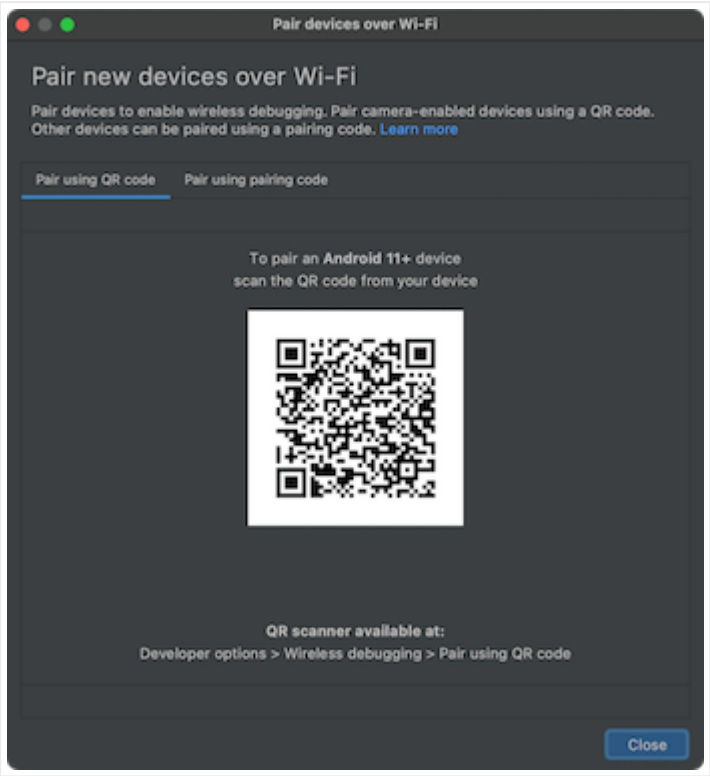


**Рис. 2.** Всплывающее окно для сопряжения устройств с помощью QR-кода или кода сопряжения.

3. На своем устройстве нажмите **«Беспроводная отладка»** и выполните сопряжение устройства:

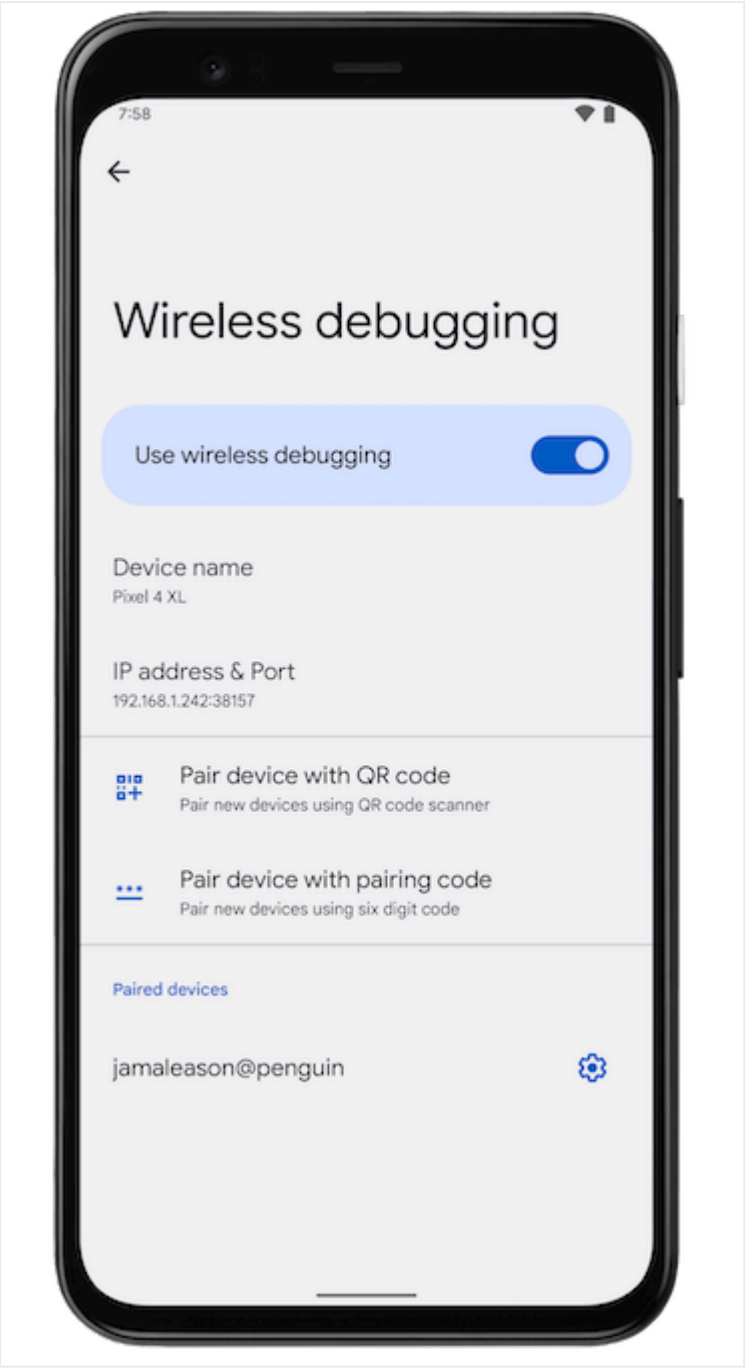


**Рис. 3.** Снимок экрана с настройками **отладки беспроводной сети** на телефоне Google Pixel.

a. Чтобы выполнить сопряжение устройства с помощью QR-кода, выберите « **Сопряжение устройства с QR-кодом»** и отсканируйте QR-код, полученный во всплывающем окне **«Сопряжение устройств через Wi-Fi»,** показанном на рисунке 2.

b. Чтобы выполнить сопряжение устройства с помощью кода сопряжения, выберите **«Сопряжение устройства с кодом сопряжения»** во всплывающем окне **«Сопряжение устройств через Wi-Fi»** . На своем устройстве выберите **«Подключиться с помощью кода сопряжения»** и запишите предоставленный шестизначный код. Как только ваше

устройство появится в окне **«Сопряжение устройств через Wi-Fi» , вы можете выбрать «Сопряжение»** и ввести шестизначный код, показанный на вашем устройстве.

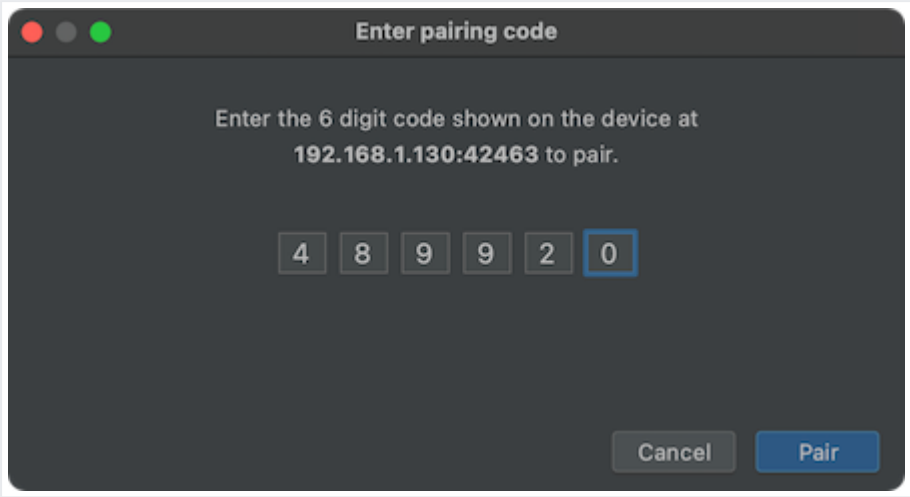


**Рисунок 4.** Пример ввода шестизначного кода.

4. После сопряжения вашего устройства вы можете попытаться развернуть приложение на своем устройстве.

   Чтобы выполнить сопряжение другого устройства или забыть текущее устройство на рабочей станции, перейдите к разделу **«Беспроводная отладка»** на своем устройстве. Коснитесь имени своей рабочей станции в разделе **«Сопряженные устройства»** и выберите **«Забыть»** .

5. Если вы хотите быстро включать и выключать беспроводную отладку, вы можете использовать плитки разработчика быстрых настроек (/studio/debug/dev-options#general) для **беспроводной отладки** , которые находятся в разделе **«Параметры разработчика»** > **«Плитки разработчика быстрых настроек»** .

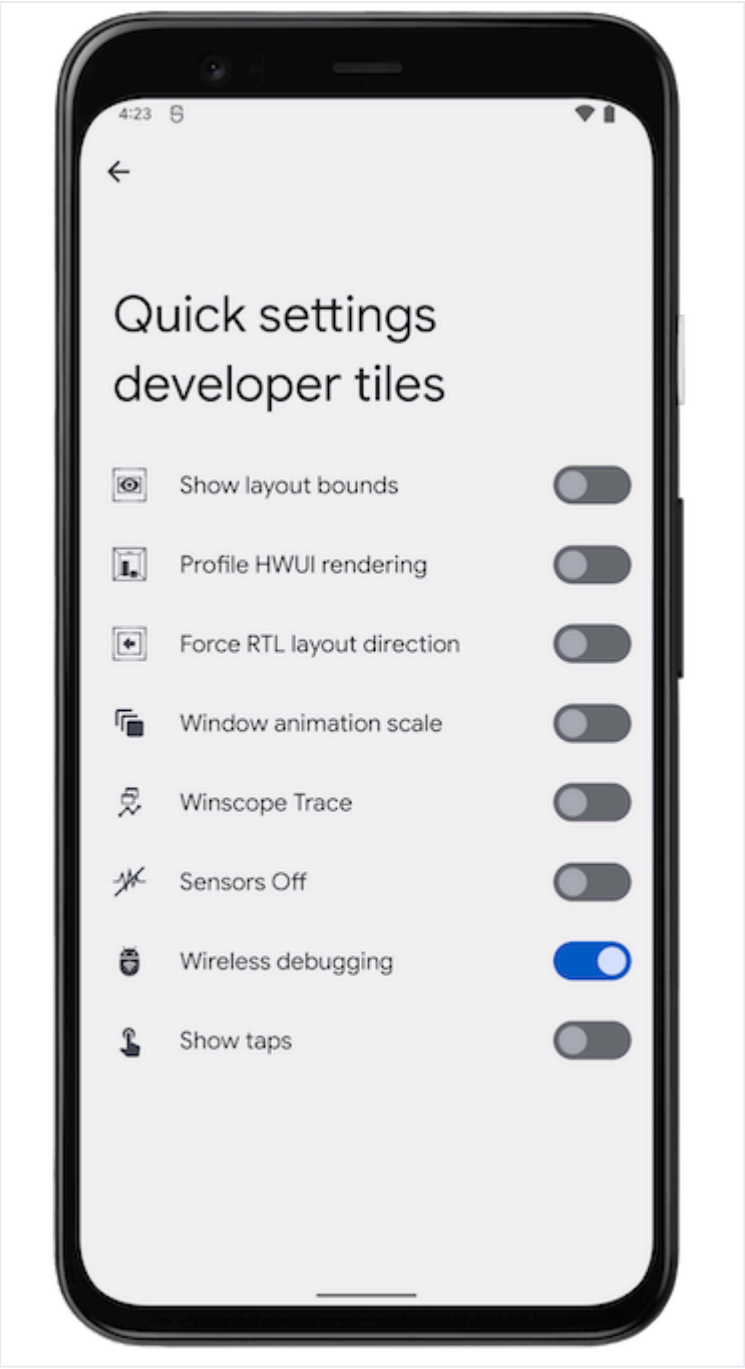


**Рис. 5.** Параметр **«Плитки разработчика быстрых настроек»** позволяет быстро включать и отключать беспроводную отладку.

## Подключение Wi-Fi с помощью командной строки

Альтернативно, чтобы подключиться к устройству с помощью командной строки без Android Studio, выполните следующие действия:

1. Включите параметры разработчика на своем устройстве, как описано ранее.
2. Включите **беспроводную отладку** на своем устройстве, как описано ранее.

3. На рабочей станции откройте окно терминала и перейдите к `android_sdk/platform-tools`.
4. Найдите свой IP-адрес, номер порта и код сопряжения, выбрав « **Сопряжение устройства с кодом сопряжения»** . Запишите IP-адрес, номер порта и код сопряжения, отображаемые на устройстве.
5. На терминале вашей рабочей станции запустите `adb pair ipaddr:port`. Используйте IP-адрес и номер порта, указанные выше.
6. При появлении запроса введите код сопряжения, как показано ниже.

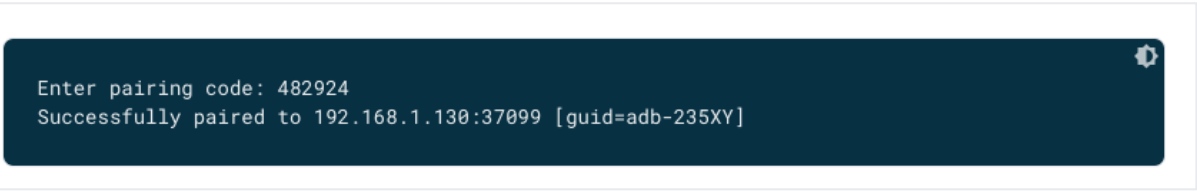


**Рисунок 6.** Сообщение указывает на то, что ваше устройство успешно сопряжено.

## Решение проблем с беспроводным подключением

Если у вас возникли проблемы с беспроводным подключением к вашему устройству, попробуйте выполнить следующие действия по устранению неполадок, чтобы решить эту проблему.

### Проверьте, соответствуют ли ваша рабочая станция и устройство предварительным требованиям.

Убедитесь, что рабочая станция и устройство соответствуют предварительным требованиям, перечисленным в начале этого раздела (#connect-to-a-device-over-wi-fi-android-11+) .

### Проверьте наличие других известных проблем

Ниже приведен список текущих известных проблем с беспроводной отладкой (с помощью adb или Android Studio) и способы их решения:

- **Wi-Fi не подключается** . Защищенные сети Wi-Fi, например корпоративные сети Wi-Fi, могут блокировать p2p-соединения и не позволять вам подключаться через Wi-Fi. Попробуйте подключиться с помощью кабеля или другой (не корпоративной) сети Wi-Fi. Беспроводное соединение через TCP/IP

(после первоначального USB-соединения) — еще один вариант, если можно использовать некорпоративную сеть.`adb connect ip:port`

- `adb`**через Wi-Fi иногда автоматически отключается** . Это может произойти, если устройство переключает сети Wi-Fi или отключается от сети. Чтобы устранить проблему, повторно подключитесь к сети.
- **Устройство не подключается после успешного сопряжения** : `adb`использование mDNS для обнаружения и автоматического подключения к сопряженным устройствам. Если ваша сеть или конфигурация устройства не поддерживает mDNS или отключила его, вам необходимо вручную подключиться к устройству с помощью .`adb connect ip:port`

## Подключитесь к устройству по беспроводной сети после первоначального подключения USB (только вариант доступен на Android 10 и более ранних версиях)

**Примечание.** Этот рабочий процесс применим также к Android 11 (и более поздним версиям), но с оговоркой, что он также предполагает *начальное* соединение через физический USB.

**Примечание.** Следующие инструкции не применимы к устройствам Wear под управлением Android 10 (уровень API 29) или ниже. Дополнительную информацию см. в руководстве по отладке приложения Wear OS . (/training/wearables/get-started/debugging#wifi-debugging)

`adb`обычно обменивается данными с устройством через USB, но вы также можете использовать `adb`Wi-Fi. Чтобы подключить устройство под управлением Android 10 (уровень API 29) или ниже, выполните следующие начальные шаги через USB:

1. Подключите свое Android-устройство и `adb`хост-компьютер к общей сети Wi-Fi.

★ **Примечание.** Помните, что не все точки доступа подходят. Возможно, вам придется использовать точку доступа, брандмауэр которой правильно настроен для поддержки **adb**.

2. Подключите устройство к главному компьютеру с помощью USB-кабеля.
3. Настройте целевое устройство на прослушивание соединения TCP/IP через порт 5555:

   ```
   adb tcpip 5555
   ```

4. Отсоедините USB-кабель от целевого устройства.
5. Найдите IP-адрес устройства Android. Например, на устройстве Nexus IP-адрес можно найти в **меню «Настройки»** > **«О планшете»** (или **«О телефоне»** ) > **«Состояние»** > **«IP-адрес»** .
6. Подключитесь к устройству по его IP-адресу:

   ```
   adb connect device_ip_address:5555
   ```

7. Убедитесь, что ваш главный компьютер подключен к целевому устройству:

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

Ваше устройство теперь подключено к `adb`.

Если `adb`соединение с вашим устройством потеряно:

- Убедитесь, что ваш хост по-прежнему подключен к той же сети Wi-Fi, что и ваше устройство Android.
- Подключитесь повторно, выполнив `adb connect`шаг еще раз.
- Если это не сработает, перезагрузите хост `adb`:

  ```
  adb kill-server
  ```

  Затем начните все сначала.

## Запрос устройств

Прежде чем давать `adb` команды, полезно знать, какие экземпляры устройств подключены к `adb` серверу. Сформируйте список подключенных устройств с помощью `devices` команды:

```
adb devices -l
```

В ответ `adb` печатает следующую информацию о состоянии для каждого устройства:

- **Серийный номер:** `adb` создает строку для уникальной идентификации устройства по номеру порта. Вот пример серийного номера: `emulator-5554`
- **Состояние:** состояние подключения устройства может быть одним из следующих:
  - `offline`: Устройство не подключено `adb` или не отвечает.
  - `device`: Устройство подключено к `adb` серверу. Обратите внимание, что это состояние не означает, что система Android полностью загружена и работает, поскольку устройство подключается, `adb` пока система еще загружается. После загрузки это нормальное рабочее состояние устройства.
  - `no device`: Устройство не подключено.
- **Описание:** Если вы включите эту `-l` опцию, `devices` команда сообщит вам, что это за устройство. Эта информация полезна, если у вас подключено несколько устройств, чтобы вы могли отличить их друг от друга.

В следующем примере показана `devices` команда и ее выходные данные. Работают три устройства. Первые две строки в списке — это эмуляторы, а третья строка — аппаратное устройство, подключаемое к компьютеру.

```
$ adb devices
List of devices attached
emulator-5556 device product:sdk_google_phone_x86_64 model:Android_SDK_built_
emulator-5554 device product:sdk_google_phone_x86 model:Android_SDK_built_for
0a388e93      device usb:1-1 product:razor model:Nexus_7 device:flo
```

## Эмулятора нет в списке

Команда `adb devices` имеет последовательность команд в крайнем случае, из-за которой запущенные эмуляторы не отображаются в `adb devices` выходных данных,

даже если эмуляторы видны на вашем рабочем столе. Это происходит, когда выполняются *все* следующие условия:

- Сервер `adb` не работает.
- Вы используете `emulator` команду с опцией `-port` или `-ports` со значением порта с нечетным номером от 5554 до 5584.
- Выбранный вами нечетный порт не занят, поэтому подключение к порту можно выполнить по указанному номеру порта — или, если он занят, эмулятор переключается на другой порт, соответствующий требованиям пункта 2.
- Вы запускаете `adb` сервер после запуска эмулятора.

Один из способов избежать этой ситуации — позволить эмулятору выбирать собственные порты и запускать не более 16 эмуляторов одновременно. Другой способ — всегда запускать `adb` сервер перед использованием `emulator` команды, как описано в следующих примерах.

**Пример 1.** В следующей последовательности команд `adb devices` команда запускает `adb` сервер, но список устройств не отображается.

Остановите `adb` сервер и введите следующие команды в указанном порядке. В качестве имени AVD укажите допустимое имя AVD из вашей системы. Чтобы получить список имен AVD, введите `emulator -list-avds`. Команда `emulator` находится в `android_sdk/tools` каталоге.

```
$ adb kill-server
$ emulator -avd Nexus_6_API_25 -port 5555
$ adb devices

List of devices attached
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
```

**Пример 2.** В следующей последовательности команд `adb devices` отображается список устройств, поскольку `adb` сервер был запущен первым.

Чтобы увидеть эмулятор в `adb devices` выходных данных, остановите `adb` сервер, а затем запустите его снова после использования команды `emulator` и перед ее использованием `adb devices`, как показано ниже:

```
$ adb kill-server
$ emulator -avd Nexus_6_API_25 -port 5557
```

```
$ adb start-server
$ adb devices

List of devices attached
emulator-5557 device
```

Дополнительные сведения о параметрах командной строки эмулятора см. в разделе Параметры запуска командной строки (/studio/run/emulator-commandline#startup-options) .

## Отправка команд на определенное устройство

Если запущено несколько устройств, при вводе `adb` команды необходимо указать целевое устройство. Чтобы указать цель, выполните следующие действия:

1. Используйте `devices` команду, чтобы получить серийный номер цели.
2. Получив серийный номер, используйте `-s` опцию с `adb` командами для указания серийного номера.
    a. Если вы собираетесь вводить много `adb` команд, вы можете `$ANDROID_SERIAL` вместо этого установить переменную среды, содержащую серийный номер.
    b. Если вы используете оба `-s` и `$ANDROID_SERIAL`, `-s` переопределяет `$ANDROID_SERIAL`.

В следующем примере получается список подключенных устройств, а затем серийный номер одного из устройств используется для установки `helloWorld.apk` на это устройство:

```
$ adb devices
List of devices attached
emulator-5554 device
emulator-5555 device
0.0.0.0:6520  device

# To install on emulator-5555
$ adb -s emulator-5555 install helloWorld.apk
# To install on 0.0.0.0:6520
$ adb -s 0.0.0.0:6520 install helloWorld.apk
```

**Примечание.** Если вы вводите команду без указания целевого устройства, когда доступно несколько устройств, **adb**отображается ошибка «adb: более одного устройства/эмулятора».

Если у вас доступно несколько устройств, но только одно из них является эмулятором, используйте `-e`опцию для отправки команд в эмулятор. Если имеется несколько устройств, но подключено только одно аппаратное устройство, используйте `-d`опцию для отправки команд на аппаратное устройство.

## Установить приложение

Вы можете использовать `adb`для установки APK на эмулятор или подключенное устройство с помощью `install`команды:

```
adb install path_to_apk
```

Вы должны использовать эту `-t`опцию с `install` командой при установке тестового APK. Для получения дополнительной информации см `-t` (#-t-option).

Чтобы установить несколько APK, используйте `install-multiple`. Это полезно, если вы загружаете все APK-файлы своего приложения для определенного устройства из Play Console и хотите установить их на эмулятор или физическое устройство.

Дополнительные сведения о том, как создать APK-файл, который можно установить на экземпляре эмулятора или устройства, см. в разделе Создание и запуск приложения (/studio/run) .

**Примечание.** Если вы используете Android Studio, вам не нужно использовать ее **adb**непосредственно для установки приложения на эмулятор или устройство. Вместо этого Android Studio самостоятельно упаковывает и устанавливает приложение.

## Настроить переадресацию портов

Используйте эту `forward`команду, чтобы настроить переадресацию произвольных портов, которая перенаправляет запросы с определенного порта хоста на другой

порт устройства. В следующем примере настраивается переадресация порта хоста 6100 на порт устройства 7100:

```
adb forward tcp:6100 tcp:7100
```

В следующем примере настраивается переадресация порта хоста 6100 на local:logd:

```
adb forward tcp:6100 local:logd
```

Это может быть полезно, если вы пытаетесь определить, что отправляется на определенный порт устройства. Все полученные данные будут записаны в демон системного журналирования и отображены в журналах устройства.

## Копирование файлов на устройство и с него

Используйте команды `pull`и `push`для копирования файлов на устройство и с него. В отличие от `install`команды, которая копирует APK-файл только в определенное место, команды `pull`и `push` позволяют копировать произвольные каталоги и файлы в любое место на устройстве.

Чтобы скопировать файл или каталог и его подкаталоги *с* устройства, выполните следующие действия:

```
adb pull remote local
```

Чтобы скопировать файл или каталог и его подкаталоги *на* устройство, выполните следующие действия:

```
adb push local remote
```

Замените `local`и `remote`на пути к целевым файлам/каталогу на вашем компьютере разработки (локальном) и на устройстве (удаленном). Например:

```
adb push myfile.txt /sdcard/myfile.txt
```

## Остановить сервер adb

В некоторых случаях вам может потребоваться завершить `adb`серверный процесс, а затем перезапустить его, чтобы решить проблему. Например, это может быть в случае, если `adb`не отвечает на команду.

Чтобы остановить `adb`сервер, используйте `adb kill-server`команду. Затем вы можете перезапустить сервер, введя любую другую `adb`команду.

## Выдать команды adb

Выдавайте `adb`команды из командной строки на вашем компьютере разработки или из сценария, используя следующее:

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

Если запущен только один эмулятор или подключено только одно устройство, `adb`команда по умолчанию отправляется на это устройство. Если запущено несколько эмуляторов и/или подключено несколько устройств, вам необходимо использовать параметр `-d`, `-e`или , `-s` чтобы указать целевое устройство, на которое должна быть направлена команда.

Вы можете просмотреть подробный список всех поддерживаемых `adb`команд, используя следующую команду:

```
adb --help
```

## Выдавать команды оболочки

Эту команду можно использовать `shell`для подачи команд устройству через `adb`интерактивную оболочку или для ее запуска. Чтобы выполнить одну команду, используйте `shell`следующую команду:

```
adb [-d |-e | -s serial_number] shell shell_command
```

Чтобы запустить интерактивную оболочку на устройстве, используйте `shell`следующую команду:

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

Чтобы выйти из интерактивной оболочки, нажмите `Control+D`или введите `exit`.

Android предоставляет большинство обычных инструментов командной строки Unix. Чтобы просмотреть список доступных инструментов, используйте следующую команду:

```
adb shell ls /system/bin
```

Справка доступна для большинства команд через `--help`аргумент. Многие команды оболочки предоставляются toybox (http://landley.net/toybox/) . Общая справка, применимая ко всем командам ToyBox, доступна через `toybox --help`.

В инструментах платформы Android 23 и более поздних версий `adb`аргументы обрабатываются так же, как и `ssh(1)`команда. Это изменение устранило множество проблем с внедрением команд (https://en.wikipedia.org/wiki/Code_injection#Shell_injection) и позволяет безопасно выполнять команды, содержащие метасимволы (https://en.wikipedia.org/wiki/Metacharacter) оболочки , такие как `adb install Let\'sGo.apk`. Это изменение означает, что интерпретация любой команды, содержащей метасимволы оболочки, также изменилась.

Например, теперь это ошибка, поскольку одинарные кавычки ( ) проглатываются локальной оболочкой, и устройство видит . Чтобы команда работала, дважды заключите кавычки: один раз для локальной оболочки и один раз для удаленной оболочки, как вы это делаете с . Например, . `adb shell setprop key 'value'` `adb shell setprop key value` `ssh(1)` `adb shell setprop key 'value'`

См. также инструмент командной строки Logcat (/studio/command-line/logcat) , который полезен для мониторинга системного журнала.

## Менеджер активности звонков

Внутри `adb`оболочки вы можете подавать команды с помощью инструмента диспетчера активности ( `am`) для выполнения различных системных действий, таких как запуск действия, принудительная остановка процесса, трансляция намерения, изменение свойств экрана устройства и т. д.

В оболочке `am`синтаксис следующий:

```
am command
```

Вы также можете выполнить команду диспетчера активности напрямую, `adb` не входя в удаленную оболочку. Например:

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

**Таблица 1.** Доступные команды диспетчера активности

| Команда | Описание |
| --- | --- |
| `start [`*`options`*`]` *`intent`* | Запустить `Activity` (/reference/android/app/Activity) указанный *`intent`*.<br><br>См. Спецификацию аргументов намерения (#IntentSpec) .<br><br>Варианты:<br><br>• `-D`: Включить отладку.<br>• `-W`: Дождитесь завершения запуска.<br>• `--start-profiler` *`file`*: Запустите профилировщик и отправьте результаты в *`file`*.<br>• `-P` *`file`*: Вроде бы `--start-profiler`, но профилирование прекращается, когда приложение простаивает.<br>• `-R` *`count`*: повторение времени запуска занятия *`count`* . Перед каждым повторением основное действие будет завершено.<br>• `-S`: принудительно остановить целевое приложение перед началом действия.<br>• `--opengl-trace`: Включить трассировку функций OpenGL.<br>• `--user` *`user_id`* `|` `current`: укажите, от имени какого пользователя запускать; если не указано, то запускать от имени текущего пользователя. |
| `startservice [`*`options`*`]` *`intent`* | Запустите `Service` (/reference/android/app/Service) указанный *`intent`*.<br><br>См. Спецификацию аргументов намерения (#IntentSpec) .<br><br>Варианты: |

| | |
|---|---|
| | • `--user` *`user_id`* \| `current`: укажите, от имени какого пользователя запускать. Если не указано, то запускать от имени текущего пользователя. |
| `force-stop` *`package`* | Принудительно остановить все, что связано с *`package`*. |
| `kill [`*`options`*`]` *`package`* | Убейте все процессы, связанные с *`package`*. Эта команда убивает только те процессы, которые безопасно завершить и которые не повлияют на работу пользователя.<br><br>Варианты:<br><br>• `--user` *`user_id`* \| `all` \| `current`: укажите процессы пользователя, которые следует уничтожить. Если не указано, то убить все процессы пользователей. |
| `kill-all` | Убейте все фоновые процессы. |
| `broadcast [`*`options`*`]` *`intent`* | Выдать намерение трансляции.<br><br>См. Спецификацию аргументов намерения (#IntentSpec) .<br><br>Варианты:<br><br>• `[--user` *`user_id`* \| `all` \| `current]`: укажите, какому пользователю отправлять сообщение. Если не указано, то отправить всем пользователям. |
| `instrument [`*`options`*`]` *`component`* | Начните мониторинг с `Instrumentation` (/reference/android/app/Instrumentation)экземпляра. Обычно целью *`component`* является форма . *`test_package`*`/`*`runner_class`*<br><br>Варианты:<br><br>• `-r`: Распечатать необработанные результаты (в противном случае декодировать *`report_key_streamresult`*). Используйте with `[-e perf true]`для генерации необработанных выходных данных для измерения производительности.<br><br>• `-e` *`name value`*: Установите *`name`* аргумент *`value`*. Для тестировщиков распространенной формой является . `-e` *`testrunner_flag value`*`[,`*`value...`*`]`<br><br>• `-p` *`file`*: запись данных профилирования в *`file`*. |

| | |
|---|---|
| | • `-w`: Прежде чем вернуться, дождитесь завершения работы инструментов. Требуется для тестировщиков.<br>• `--no-window-animation`: отключить анимацию окон во время работы.<br>• `--user` ***`user_id`*** `\| current`: Укажите, какой пользовательский инструментарий запускается. Если не указано, запустите от текущего пользователя. |
| `profile start` ***`process file`*** | Запустите профайлер ***`process`***, запишите результаты в ***`file`***. |
| `profile stop` ***`process`*** | Остановите профайлер на ***`process`***. |
| `dumpheap [`***`options`***`]` ***`process file`*** | Сбросьте кучу ***`process`***, напишите в ***`file`***.<br>Варианты:<br>• `--user [`***`user_id`*** `\| current]`: При указании имени процесса укажите пользователя процесса, который нужно выгрузить. Если не указано, используется текущий пользователь.<br>• `-n`: Дамп собственной кучи вместо управляемой кучи. |
| `set-debug-app [`***`options`***`]` ***`package`*** | Установите приложение ***`package`***для отладки.<br>Варианты:<br>• `-w`: дождаться отладчика при запуске приложения.<br>• `--persistent`: сохранить это значение. |
| `clear-debug-app` | Очистите пакет, предыдущий набор для отладки, с помощью `set-debug-app`. |
| `monitor [`***`options`***`]` | Начните мониторинг сбоев или ошибок ANR.<br>Варианты:<br>• `--gdb`: запуск `gdbserv`на указанном порту при сбое/ANR. |
| `screen-compat {on \| off}` ***`package`*** | Режим совместимости экрана (/guide/practices/screen-compat-mode) управления |

| | |
|---|---|
| | *package*. |
| `display-size [reset \| widthxheight]` | Переопределить размер дисплея устройства. Эта команда полезна для тестирования вашего приложения на экранах разных размеров, имитируя маленькое разрешение экрана на устройстве с большим экраном, и наоборот.<br><br>Пример:<br>`am display-size 1280x800` |
| `display-density dpi` | Переопределить плотность отображения устройства. Эта команда полезна для тестирования вашего приложения на экране с различной плотностью экрана, имитируя среду экрана с высокой плотностью экрана с использованием экрана с низкой плотностью, и наоборот.<br><br>Пример:<br>`am display-density 480` |
| `to-uri intent` | Распечатайте данную спецификацию намерения как URI.<br><br>См. Спецификацию аргументов намерения (#IntentSpec) . |
| `to-intent-uri intent` | Распечатайте данную спецификацию намерения как `intent:`URI.<br><br>См. Спецификацию аргументов намерения (#IntentSpec) . |

### Спецификация аргументов намерения

Для команд диспетчера действий, которые принимают `intent`аргумент, вы можете указать намерение с помощью следующих параметров:

Показать все

`-a action`

Specify the intent action, such as `android.intent.action.VIEW`. You can declare this only once.

`-d data_uri`

Specify the intent data URI, such as `content://contacts/people/1`. You can declare this only once.

`-t *mime_type*`

Specify the intent MIME type, such as `image/png`. You can declare this only once.

`-c *category*`

Specify an intent category, such as `android.intent.category.APP_CONTACTS`.

`-n *component*`

Specify the component name with package name prefix to create an explicit intent, such as `com.example.app/.ExampleActivity`.

`-f *flags*`

Add flags to the intent, as supported by `setFlags()` (/reference/android/content/Intent#setFlags(int)).

`--esn *extra_key*`

Add a null extra. This option is not supported for URI intents.

`-e | --es *extra_key extra_string_value*`

Add string data as a key-value pair.

`--ez *extra_key extra_boolean_value*`

Add boolean data as a key-value pair.

`--ei *extra_key extra_int_value*`

Add integer data as a key-value pair.

`--el *extra_key extra_long_value*`

Add long data as a key-value pair.

`--ef *extra_key extra_float_value*`

Add float data as a key-value pair.

`--eu *extra_key extra_uri_value*`

Add URI data as a key-value pair.

`--ecn *extra_key extra_component_name_value*`

Add a component name, which is converted and passed as a `ComponentName` (/reference/android/content/ComponentName) object.

`--eia *extra_key extra_int_value*[,*extra_int_value...*]`

Add an array of integers.

`--ela *extra_key extra_long_value*[,*extra_long_value...*]`

Add an array of longs.

`--efa *extra_key extra_float_value*[,*extra_float_value...*]`

Add an array of floats.

`--grant-read-uri-permission`

Include the flag `FLAG_GRANT_READ_URI_PERMISSION` (/reference/android/content/Intent#FLAG_GRANT_READ_URI_PERMISSION).

`--grant-write-uri-permission`

Include the flag `FLAG_GRANT_WRITE_URI_PERMISSION` (/reference/android/content/Intent#FLAG_GRANT_WRITE_URI_PERMISSION).

`--debug-log-resolution`

Include the flag `FLAG_DEBUG_LOG_RESOLUTION` (/reference/android/content/Intent#FLAG_DEBUG_LOG_RESOLUTION).

`--exclude-stopped-packages`

Include the flag `FLAG_EXCLUDE_STOPPED_PACKAGES` (/reference/android/content/Intent#FLAG_EXCLUDE_STOPPED_PACKAGES).

`--include-stopped-packages`

Include the flag `FLAG_INCLUDE_STOPPED_PACKAGES` (/reference/android/content/Intent#FLAG_INCLUDE_STOPPED_PACKAGES).

`--activity-brought-to-front`

Include the flag `FLAG_ACTIVITY_BROUGHT_TO_FRONT` (/reference/android/content/Intent#FLAG_ACTIVITY_BROUGHT_TO_FRONT).

`--activity-clear-top`

Include the flag `FLAG_ACTIVITY_CLEAR_TOP` (/reference/android/content/Intent#FLAG_ACTIVITY_CLEAR_TOP).

`--activity-clear-when-task-reset`

Include the flag `FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET` (/reference/android/content/Intent#FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET).

`--activity-exclude-from-recents`

Include the flag `FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS` (/reference/android/content/Intent#FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS).

`--activity-launched-from-history`

Include the flag `FLAG_ACTIVITY_LAUNCHED_FROM_HISTORY` (/reference/android/content/Intent#FLAG_ACTIVITY_LAUNCHED_FROM_HISTORY).

`--activity-multiple-task`

Include the flag `FLAG_ACTIVITY_MULTIPLE_TASK` (/reference/android/content/Intent#FLAG_ACTIVITY_MULTIPLE_TASK).

`--activity-no-animation`

Include the flag `FLAG_ACTIVITY_NO_ANIMATION` (/reference/android/content/Intent#FLAG_ACTIVITY_NO_ANIMATION).

`--activity-no-history`

Include the flag `FLAG_ACTIVITY_NO_HISTORY` (/reference/android/content/Intent#FLAG_ACTIVITY_NO_HISTORY).

`--activity-no-user-action`

Include the flag `FLAG_ACTIVITY_NO_USER_ACTION` (/reference/android/content/Intent#FLAG_ACTIVITY_NO_USER_ACTION).

`--activity-previous-is-top`

Include the flag `FLAG_ACTIVITY_PREVIOUS_IS_TOP` (/reference/android/content/Intent#FLAG_ACTIVITY_PREVIOUS_IS_TOP).

`--activity-reorder-to-front`

Include the flag `FLAG_ACTIVITY_REORDER_TO_FRONT` (/reference/android/content/Intent#FLAG_ACTIVITY_REORDER_TO_FRONT).

`--activity-reset-task-if-needed`

Include the flag `FLAG_ACTIVITY_RESET_TASK_IF_NEEDED` (/reference/android/content/Intent#FLAG_ACTIVITY_RESET_TASK_IF_NEEDED).

`--activity-single-top`

Include the flag `FLAG_ACTIVITY_SINGLE_TOP` (/reference/android/content/Intent#FLAG_ACTIVITY_SINGLE_TOP).

`--activity-clear-task`

Include the flag `FLAG_ACTIVITY_CLEAR_TASK` (/reference/android/content/Intent#FLAG_ACTIVITY_CLEAR_TASK).

`--activity-task-on-home`

Include the flag `FLAG_ACTIVITY_TASK_ON_HOME` (/reference/android/content/Intent#FLAG_ACTIVITY_TASK_ON_HOME).

`--receiver-registered-only`

Include the flag `FLAG_RECEIVER_REGISTERED_ONLY` (/reference/android/content/Intent#FLAG_RECEIVER_REGISTERED_ONLY).

`--receiver-replace-pending`

Include the flag `FLAG_RECEIVER_REPLACE_PENDING` (/reference/android/content/Intent#FLAG_RECEIVER_REPLACE_PENDING).

`--selector`

Requires the use of `-d` and `-t` options to set the intent data and type.

*`URI component package`*

You can directly specify a URI, package name, and component name when not qualified by one of the preceding options. When an argument is unqualified, the tool assumes the argument is a URI if it contains a ":" (colon). The tools assumes the argument is a component name if it contains a "/" (forward-slash); otherwise it assumes the argument is a package name.

## Вызов менеджера пакетов ( `pm`)

Внутри `adb`оболочки вы можете подавать команды с помощью инструмента диспетчера пакетов ( `pm`), чтобы выполнять действия и запросы к пакетам приложений, установленным на устройстве.

В оболочке `pm`синтаксис следующий:

```
pm command
```

Вы также можете выполнить команду менеджера пакетов напрямую, `adb` не входя в удаленную оболочку. Например:

```
adb shell pm uninstall com.example.MyApp
```

**Таблица 2.** Доступные команды менеджера пакетов

| Команда | Описание |
| --- | --- |
| `list packages [options] filter` | Распечатайте все пакеты, при необходимости только те, имя пакета которых содержит текст в формате *`filter`*.<br><br>Параметры:<br><br>• `-f`: См. связанный файл.<br>• `-d`: фильтр для отображения только отключенных пакетов.<br>• `-e`: фильтр для отображения только включенных пакетов.<br>• `-s`: фильтр для отображения только системных пакетов.<br>• `-3`: фильтр для отображения только сторонних пакетов.<br>• `-i`: пакеты смотрите в установщике.<br>• `-u`: включить неустановленные пакеты.<br>• `--user user_id`: Пространство пользователя для запроса. |

| | |
|---|---|
| `list permission-groups` | Распечатайте все известные группы разрешений. |
| `list permissions [`*`options`*`] `*`group`* | Распечатайте все известные разрешения, при необходимости только те, которые находятся в формате *`group`*.<br><br>Параметры:<br><br>• `-g`: Организовать по группам.<br>• `-f`: Распечатать всю информацию.<br>• `-s`: Краткое содержание.<br>• `-d`: перечислять только опасные разрешения.<br>• `-u`: перечислить только те разрешения, которые будут видеть пользователи. |
| `list instrumentation [`*`options`*`]` | Перечислите все тестовые пакеты.<br><br>Параметры:<br><br>• `-f`: укажите APK-файл тестового пакета.<br>• *`target_package`*: список тестовых пакетов только для этого приложения. |
| `list features` | Распечатайте все характеристики системы. |
| `list libraries` | Распечатайте все библиотеки, поддерживаемые текущим устройством. |
| `list users` | Распечатайте всех пользователей в системе. |
| `path `*`package`* | Выведите путь к APK данного файла *`package`*. |
| `install [`*`options`*`] `*`path`* | *`path`*Установите в систему пакет, указанный .<br><br>Параметры:<br><br>• `-r`: переустановить существующее приложение, сохранив его данные.<br>• `-t`: разрешить установку тестовых APK. Gradle генерирует тестовый APK-файл, когда вы только запустили или отладили свое приложение или |

использовали команду Android Studio **Build > Build APK** . Если APK создан с использованием SDK предварительной версии для разработчиков, необходимо включить эту `-t`опцию (/studio/command-line/adb#-t-option) в `install`команду, если вы устанавливаете тестовый APK.

- `-i` *`installer_package_name`*: укажите имя установочного пакета.
- `--install-location` *`location`*: укажите место установки, используя одно из следующих значений:
  - `0`: использовать папку установки по умолчанию.
  - `1`: Установить на внутреннюю память устройства.
  - `2`: Установить на внешний носитель.
- `-f`: Установить пакет во внутреннюю системную память.
- `-d`: Разрешить понижение версии кода.
- `-g`: предоставить все разрешения, перечисленные в манифесте приложения.
- `--fastdeploy`: быстро обновите установленный пакет, обновив только измененные части APK.
- `--incremental`: устанавливает достаточное количество APK для запуска приложения с одновременной потоковой передачей оставшихся данных в фоновом режиме. Чтобы использовать эту функцию, необходимо подписать APK, создать файл схемы подписи APK v4 (/studio/command-line/apksigner#v4-signing-enabled) и поместить этот файл в тот же каталог, что и APK. Эта функция поддерживается только на некоторых устройствах. Эта опция заставляет `adb`использовать эту функцию или отказать, если она не поддерживается, с подробной информацией о том, почему она не удалась. Добавьте `--wait`возможность дождаться полной установки APK, прежде чем предоставлять доступ к APK.

  `--no-incremental`препятствует `adb`использованию этой функции.

| | |
|---|---|
| `uninstall [`*`options`*`] `*`package`* | Удаляет пакет из системы.<br><br>Параметры:<br><br>• `-k`: сохранить каталоги данных и кеша после удаления пакета.<br>• `--user `*`user_id`*: указывает пользователя, для которого удаляется пакет.<br>• `--versionCode `*`version_code`*: удаление выполняется только в том случае, если приложение имеет указанный код версии. |
| `clear `*`package`* | Удалить все данные, связанные с пакетом. |
| `enable `*`package_or_component`* | Включите данный пакет или компонент (записанный как «пакет/класс»). |
| `disable `*`package_or_component`* | Отключите данный пакет или компонент (записанный как «пакет/класс»). |
| `disable-user [`*`options`*`] `*`package_or_component`* | Параметры:<br><br>• `--user `*`user_id`*: Пользователь, которого нужно отключить. |
| `grant `*`package_name permission`* | Предоставьте разрешение приложению. На устройствах под управлением Android 6.0 (уровень API 23) и выше разрешением может быть любое разрешение, объявленное в манифесте приложения. На устройствах под управлением Android 5.1 (уровень API 22) и ниже должно быть дополнительное разрешение, определенное приложением. |
| `revoke `*`package_name permission`* | Отозвать разрешение у приложения. На устройствах под управлением Android 6.0 (уровень API 23) и выше разрешением может быть любое разрешение, объявленное в манифесте приложения. На устройствах под управлением Android 5.1 (уровень API 22) и ниже должно быть дополнительное разрешение, определенное приложением. |
| `set-install-location `*`location`* | Измените место установки по умолчанию. Значения местоположения: |

| | |
|---|---|
| | • `0`: Авто: Пусть система сама выберет лучшее место.<br>• `1`: Внутренний: Установить на внутреннюю память устройства.<br>• `2`:Внешний: Установить на внешний носитель.<br><br>★ **Примечание.** Это предназначено только для отладки. Использование этого может привести к сбою приложений и другому нежелательному поведению. |
| `get-install-location` | Возвращает текущее место установки. Возвращаемые значения:<br>• `0 [auto]`: позвольте системе выбрать лучшее место.<br>• `1 [internal]`: Установить на внутреннюю память устройства.<br>• `2 [external]`: Установить на внешний носитель |
| `set-permission-enforced` *`permission`* `[true \| false]` | Укажите, следует ли применять данное разрешение. |
| `trim-caches` *`desired_free_space`* | Обрезать файлы кэша, чтобы достичь заданного свободного места. |
| `create-user` *`user_name`* | Создайте нового пользователя с заданным значением , распечатав новый идентификатор пользователя. *`user_name`* |
| `remove-user` *`user_id`* | Удалить пользователя с указанным пользователем , удалив все данные, связанные с этим пользователем. *`user_id`* |
| `get-max-users` | Распечатайте максимальное количество пользователей, поддерживаемых устройством. |
| `get-app-links [`*`options`*`] [`*`package`*`]` | Распечатайте состояние проверки домена для данного*`package`*или для всех пакетов, если ни один не указан. Коды штатов определяются следующим образом: |

| | |
|---|---|
| | • `none`: для этого домена ничего не записано<br>• `verified`: домен успешно подтвержден<br>• `approved`: принудительное одобрение, обычно через оболочку<br>• `denied`: принудительно запрещено, обычно через оболочку<br>• `migrated`: сохранена проверка из устаревшего ответа<br>• `restored`: сохраненная проверка после восстановления пользовательских данных<br>• `legacy_failure`: отклонено устаревшим проверяющим по неизвестной причине<br>• `system_configured`: автоматически утверждается конфигурацией устройства<br>• `>= 1024`: собственный код ошибки, специфичный для верификатора устройства.<br><br>Варианты:<br><br>• `--user` *`user_id`*: включить выбор пользователя. Включите все домены, а не только автопроверку. |
| `reset-app-links [`*`options`*`] [`*`package`*`]` | Сбросить состояние проверки домена для данного пакета или для всех пакетов, если ни один не указан.<br><br>• *`package`*: пакет для сброса или «все», чтобы сбросить все пакеты.<br><br>Варианты:<br><br>• `--user` *`user_id`*: включить выбор пользователя. Включите все домены, а не только автопроверку. |
| `verify-app-links [`*`--re-verify`*`] [`*`package`*`]` | Отправьте запрос на проверку для данного*`package`*или для всех пакетов, если ни один не указан. Отправляется только в том случае, если пакет ранее не записал ответ.<br><br>• `--re-verify`: отправить, даже если пакет записал ответ |

| | |
|---|---|
| `set-app-links [--package package] state domains` | Вручную установите состояние домена для пакета. Чтобы это работало, домен должен быть объявлен пакетом как autoVerify. Эта команда не будет сообщать об ошибке для доменов, которые невозможно применить.<br>• `--package package`: пакет для установки или «все», чтобы установить все пакеты.<br>• `state`: код для установки доменов. Допустимые значения:<br>  • `STATE_NO_RESPONSE (0)`: сброс, как если бы ответ не был записан.<br>  • `STATE_SUCCESS (1)`: считать домен успешно проверенным агентом проверки домена. Обратите внимание, что агент проверки домена может отменить это.<br>  • `STATE_APPROVED (2)`: рассматривать домен как всегда одобренный, не позволяя агенту проверки домена изменить его.<br>  • `STATE_DENIED (3)`: рассматривать домен как всегда отклоненный, не позволяя агенту проверки домена изменить его.<br>• `domains`: разделенный пробелами список доменов, которые нужно изменить, или «все», чтобы изменить каждый домен. |
| `set-app-links-user-selection --user user_id [--package package] enabled domains` | Вручную установите состояние выбора пользователя хоста для пакета. Чтобы это работало, домен должен быть объявлен пакетом. Эта команда не будет сообщать об ошибке для доменов, которые невозможно применить.<br>• `--user user_id`: пользователь, для которого можно изменить выбор<br>• `--package package`: пакет для установки<br>• `enabled`: утверждать ли домен<br>• `domains`: список доменов, разделенных пробелами, которые нужно изменить, или «все», чтобы изменить каждый домен. |

| | |
|---|---|
| `set-app-links-user-selection --user` *`user_id`* `[--package` *`package`*`]` *`enabled domains`* | Вручную установите состояние выбора пользователя хоста для пакета. Чтобы это работало, домен должен быть объявлен пакетом. Эта команда не будет сообщать об ошибке для доменов, которые невозможно применить.<br>• `--user` *`user_id`*: пользователь, для которого можно изменить выбор<br>• `--package` *`package`*: пакет для установки<br>• *`enabled`*: утверждать ли домен<br>• *`domains`*: список доменов, разделенных пробелами, которые нужно изменить, или «все», чтобы изменить каждый домен. |
| `set-app-links-allowed --user` *`user_id`* `[--package` *`package`*`]` *`allowed`* | Переключите настройку автоматической проверки ссылок для пакета.<br>• `--user` *`user_id`*: пользователь, для которого можно изменить выбор<br>• `--package` *`package`*: пакет для установки или «все», чтобы установить все пакеты; пакеты будут сброшены, если пакет не указан<br>• *`allowed`*: true, чтобы пакет мог открывать автоматически проверенные ссылки, false, чтобы отключить |
| `get-app-link-owners --user` *`user_id`* `[--package` *`package`*`]` *`domains`* | Выведите владельцев определенного домена для данного пользователя в порядке от низкого к высокому приоритету.<br>• `--user` *`user_id`*: пользователь, которого нужно запросить<br>• `--package` *`package`*: дополнительно также печатать для всех веб-доменов, объявленных пакетом, или «все», чтобы печатать все пакеты.<br>• *`domains`*: список доменов, разделенных пробелами, для запроса |

## Вызов диспетчера политики устройств ( `dpm`)

Чтобы помочь вам разработать и протестировать приложения для управления устройствами, дайте команды `dpm`инструменту диспетчера политик устройств ( ). Используйте этот инструмент для управления активным приложением администратора или изменения данных о состоянии политики на устройстве.

В оболочке `dpm`синтаксис следующий:

```
dpm command
```

Вы также можете выполнить команду диспетчера политики устройств напрямую, `adb` не входя в удаленную оболочку:

```
adb shell dpm command
```

**Таблица 3.** Доступные команды диспетчера политик устройств

| Команда | Описание |
| --- | --- |
| `set-active-admin [options] component` | Наборы*component*как активный администратор<br>Варианты:<br>• `--user user_id`: Укажите целевого пользова |
| `set-profile-owner [options] component` | Набор*component*в качестве активного администр<br>Варианты:<br>• `--user user_id`: Укажите целевого пользова<br>• `--name name`: укажите удобочитаемое назва |
| `set-device-owner [options] component` | Набор*component*как активный администратор и<br>Варианты:<br>• `--user user_id`: Укажите целевого пользова<br>• `--name name`: укажите удобочитаемое назва |
| `remove-active-admin [options] component` | Отключить активного администратора. Приложе в манифесте. Эта команда также удаляет владел<br>Варианты: |

| | |
|---|---|
| | • `--user` *`user_id`*: Укажите целевого пользова |
| `clear-freeze-period-record` | Очистите запись устройства о ранее установлен ограничений планирования устройства при разр обновлениями системы (/work/dpc/system-update<br><br>Поддерживается на устройствах под управлени |
| `force-network-logs` | Заставьте систему подготовить все существующ получает `onNetworkLogsAvailable()` (/reference/android/app/admin/DeviceAdminRecei обратный вызов. См. Журнал сетевой активност<br><br>Эта команда ограничена по скорости. Поддержи |
| `force-security-logs` | Заставьте систему сделать все существующие ж `onSecurityLogsAvailable()` (/reference/android/app/admin/DeviceAdminRecei вызов. См. Журнал активности корпоративных у<br><br>Эта команда ограничена по скорости. Поддержи |

## Сделать снимок экрана

Команда `screencap` представляет собой утилиту оболочки для создания снимка экрана устройства.

В оболочке `screencap` синтаксис следующий:

```
screencap filename
```

Для использования `screencap` из командной строки введите следующее:

```
adb shell screencap /sdcard/screen.png
```

Вот пример сеанса создания снимков экрана с использованием `adb` оболочки для создания снимка экрана и `pull` команды для загрузки файла с устройства:

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

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

## Записать видео

Команда `screenrecord`представляет собой утилиту оболочки для записи дисплея устройств под управлением Android 4.4 (уровень API 19) и выше. Утилита записывает действия на экране в файл MPEG-4. Вы можете использовать этот файл для создания рекламных или обучающих видеороликов, а также для отладки и тестирования.

В оболочке используйте следующий синтаксис:

```
screenrecord [options] filename
```

Для использования `screenrecord`из командной строки введите следующее:

```
adb shell screenrecord /sdcard/demo.mp4
```

Остановите запись экрана, нажав Control+C. В противном случае запись автоматически останавливается через три минуты или по истечении времени, установленного параметром `--time-limit`.

Чтобы начать запись экрана вашего устройства, выполните `screenrecord`команду для записи видео. Затем запустите `pull`команду для загрузки видео с устройства на главный компьютер. Вот пример сеанса записи:

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

Утилита `screenrecord`может записывать с любым поддерживаемым разрешением и битрейтом, которые вы запрашиваете, сохраняя при этом соотношение сторон дисплея устройства. По умолчанию утилита записывает с исходным разрешением и ориентацией дисплея, максимальная продолжительность — три минуты.

Ограничения утилиты `screenrecord`:

- Звук не записывается вместе с видеофайлом.

- Запись видео недоступна для устройств под управлением Wear OS.
- Некоторые устройства могут не поддерживать запись с исходным разрешением экрана. Если у вас возникли проблемы с записью экрана, попробуйте использовать более низкое разрешение экрана.
- Поворот экрана во время записи не поддерживается. Если экран поворачивается во время записи, часть экрана при записи обрезается.

**Таблица 4.** `screenrecord` варианты

| Параметры | Описание |
|---|---|
| `--help` | Отобразить синтаксис и параметры команды |
| `--size` *`width`*`x`*`height`* | Установите размер видео: `1280x720`. Значением по умолчанию является собственное разрешение экрана устройства (если оно поддерживается), в противном случае — 1280x720. Для достижения наилучших результатов используйте размер, поддерживаемый кодером Advanced Video Coding (AVC) вашего устройства. |
| `--bit-rate` *`rate`* | Установите битрейт видео в мегабитах в секунду. Значение по умолчанию — 4 Мбит/с. Вы можете увеличить скорость передачи данных, чтобы улучшить качество видео, но это приведет к увеличению размера файлов фильмов. В следующем примере устанавливается скорость записи 6 Мбит/с:<br><br>`screenrecord --bit-rate 6000000 /sdcard/demo.mp4` |
| `--time-limit` *`time`* | Установите максимальное время записи в секундах. Значение по умолчанию и максимальное значение — 180 (3 минуты). |
| `--rotate` | Поверните выход на 90 градусов. Эта функция является экспериментальной. |
| `--verbose` | Отображение информации журнала на экране командной строки. Если вы не установите эту опцию, утилита не будет отображать никакой информации во время работы. |

## Чтение профилей ART для приложений

Начиная с Android 7.0 (уровень API 24), среда выполнения Android (ART) собирает профили выполнения установленных приложений, которые используются для оптимизации производительности приложений. Изучите собранные профили, чтобы понять, какие методы выполняются часто и какие классы используются во время запуска приложения.

**Примечание.** Получить имя файла профиля выполнения можно только в том случае, если у вас есть root-доступ к файловой системе, например, в эмуляторе.

Чтобы создать текстовую форму информации профиля, используйте следующую команду:

```
adb shell cmd package dump-profiles package
```

Чтобы получить созданный файл, используйте:

```
adb pull /data/misc/profman/package.prof.txt
```

## Сброс тестовых устройств

Если вы тестируете свое приложение на нескольких тестовых устройствах, может оказаться полезным выполнять сброс настроек вашего устройства между тестами, например, чтобы удалить пользовательские данные и сбросить тестовую среду. Вы можете выполнить сброс настроек тестового устройства под управлением Android 10 (уровень API 29) или выше с помощью `testharness` `adb`команды оболочки, как показано:

```
adb shell cmd testharness enable
```

При восстановлении устройства с помощью `testharness`устройства автоматически создается резервная копия ключа RSA, который позволяет выполнять отладку на текущей рабочей станции в постоянном расположении. То есть после перезагрузки устройства рабочая станция может продолжать отладку и выдавать `adb`команды устройству без ручной регистрации нового ключа.

Кроме того, чтобы упростить и повысить безопасность тестирования вашего приложения, использование `testharness`для восстановления устройства также

изменяет следующие настройки устройства:

- В устройстве настраиваются определенные системные параметры, чтобы не появлялись мастера первоначальной настройки устройства. То есть устройство переходит в состояние, из которого вы можете быстро установить, отладить и протестировать свое приложение.
- Настройки:
  - Отключает экран блокировки.
  - Отключает экстренные оповещения.
  - Отключает автоматическую синхронизацию учетных записей.
  - Отключает автоматическое обновление системы.
- Другой:
  - Отключает предустановленные приложения безопасности.

Если вашему приложению необходимо обнаружить и адаптироваться к настройкам команды по умолчанию `testharness` , используйте файл . `ActivityManager.isRunningInUserTestHarness()` (/reference/android/app/ActivityManager#isRunningInUserTestHarness())

## Склайт

`sqlite3`запускает `sqlite`программу командной строки для проверки баз данных SQLite. Он включает в себя такие команды, как `.dump`печать содержимого таблицы и `.schema`вывод `SQL CREATE`оператора для существующей таблицы. Вы также можете выполнять команды SQLite из командной строки, как показано:

```
$ adb -s emulator-5554 shell
$ sqlite3 /data/data/com.example.app/databases/rssitems.db
SQLite version 3.3.12
Enter ".help" for instructions
```

**Примечание.** Доступ к базе данных SQLite возможен только при наличии root-доступа к файловой системе, например, в эмуляторе.

Дополнительную информацию см. `sqlite3`в документации по командной строке (http://www.sqlite.org/cli.html) .

## USB-серверы adb

Сервер adb может взаимодействовать со стеком USB через два бэкэнда. Он может использовать либо собственный бэкэнд ОС (Windows, Linux или macOS), либо бэкенд `libusb`. Некоторые функции, такие как `attach`, `detach`и определение скорости USB, доступны только при использовании `libusb`серверной части.

Вы можете выбрать серверную часть, используя `ADB_LIBUSB`переменную среды. Если он не установлен, adb использует свой бэкэнд по умолчанию. Поведение по умолчанию варьируется в зависимости от ОС. Начиная с уровня API 34, по умолчанию используется собственный бэкэнд. Если этот параметр установлен, он определяет , используется `ADB_LIBUSB`ли собственный бэкэнд или нет. Дополнительную информацию о переменных среды adb `libusb`см . на странице руководства adb.
(https://android.googlesource.com/platform/packages/modules/adb/+/refs/heads/master/docs/user/adb.1.md)

**Экспериментально:** поддержка использования **`libusb`**серверной части с Windows является экспериментальной. Начиная с уровня API 34, с библиотекой были протестированы только платформы macOS и Linux **`libusb`**.

## бэкэнды adb mDNS

ADB может использовать протокол многоадресной рассылки DNS для автоматического подключения сервера и устройств. Сервер ADB поставляется с двумя серверными модулями: Bonjour (mdnsResponder от Apple) и Openscreen.

Серверной части Bonjour необходим демон, работающий на хост-компьютере. В macOS встроенный демон Apple всегда работает, но в Windows и Linux пользователь должен убедиться, что `mdnsd`демон запущен и работает. Если команда `adb mdns check`возвращает ошибку, вполне вероятно, что ADB использует серверную часть Bonjour, но демон Bonjour не запущен.

Серверной части Openscreen не требуется запуск демона на компьютере. Поддержка серверной части Openscreen в macOS начинается с ADB v35. Windows и Linux поддерживаются начиная с ADB v34.

По умолчанию ADB использует серверную часть Bonjour. Это поведение можно изменить с помощью переменной среды `ADB_MDNS_OPENSCREEN` (установленной в

`1` или `0`). Дополнительную информацию см. на странице руководства АБР . (https://android.googlesource.com/platform/packages/modules/adb/+/refs/heads/master/docs/user/adb.1.md)



Последнее обновление: 9 февраля 2024 г. по всемирному координированному времени.