Запись и просмотр журналов с Logcat

Окно **Logcat** в Android Studio отображает системные сообщения, например, когда происходит сборка мусора, и сообщения, которые вы добавили в свое приложение с помощью [Log](https://developer.android.com/reference/android/util/Log.html)класса. Он отображает сообщения в режиме реального времени и хранит историю, чтобы вы могли просматривать старые сообщения.

Чтобы отображать только интересующую вас информацию, вы можете создавать фильтры, изменять объем отображаемой информации в сообщениях, устанавливать уровни приоритетов, отображать сообщения, созданные только кодом приложения, и выполнять поиск в журнале. По умолчанию logcat показывает выходные данные журнала, относящиеся только к последнему запущенному приложению.

Когда приложение выдает исключение, logcat показывает сообщение, за которым следует связанная трассировка стека, содержащая ссылки на строку кода.

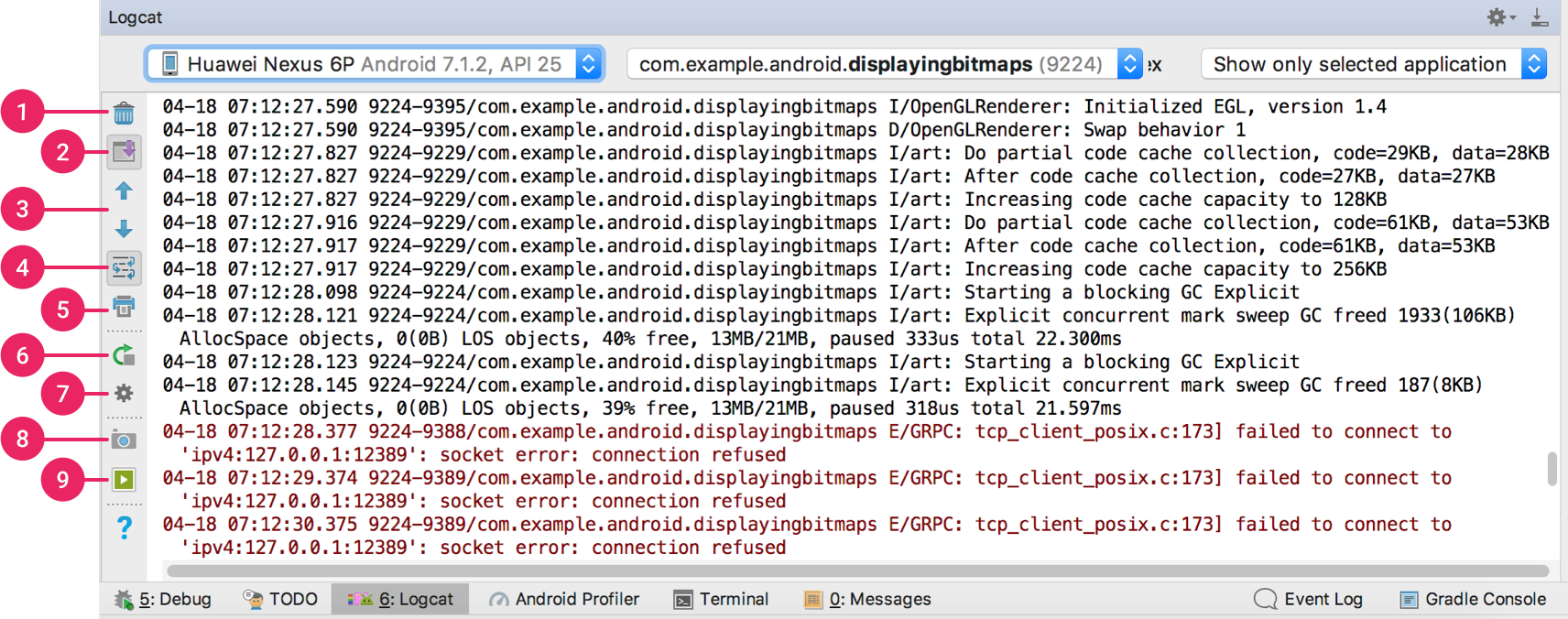
Начиная с Android Studio 2.2, в окне « **Выполнить»** также отображаются сообщения журнала для текущего запущенного приложения. Обратите внимание, что вы можете настроить отображение вывода logcat, но не окно **Run** .

Просмотр журналов вашего приложения

Чтобы отобразить сообщения журнала для приложения:

1. [Создайте и запустите ваше приложение](https://developer.android.com/studio/run/index.html#RunningApp) на устройстве.
2. Нажмите **Вид> Окно инструментов> Logcat** (или нажмите **Logcat** https://developer.android.com/studio/images/buttons/logcat-icon.png в **панели** инструментов).

Окно Logcat показывает сообщения журнала для выбранного приложения, которые выбраны из выпадающих списков в верхней части окна, как показано на рисунке 1.



**Рисунок 1.** Окно Logcat

По умолчанию logcat отображает только сообщения журнала для вашего приложения, работающего на устройстве. Чтобы изменить это значение по умолчанию, посмотрите, как [фильтровать сообщения logcat](https://developer.android.com/studio/debug/am-logcat#filtering) .

Панель инструментов Logcat содержит следующие кнопки:

1. **Очистить logcat** https://developer.android.com/studio/images/buttons/logcat-clear.png : Нажмите, чтобы очистить видимый журнал.
2. **Прокрутить до конца** https://developer.android.com/studio/images/buttons/logcat-scroll-end.png : нажмите, чтобы перейти к нижней части журнала и увидеть последние сообщения журнала. Если затем щелкнуть строку в журнале, представление приостановится в этой точке.
3. **Вверх по трассе стека** https://developer.android.com/studio/images/buttons/logcat-trace-up.png и **Вниз по трассе стека** https://developer.android.com/studio/images/buttons/logcat-trace-down.png : щелкните, чтобы перемещаться вверх и вниз по трассе стека в журнале, выбирая последующие имена файлов (и просматривая соответствующие номера строк в редакторе), которые появляются в напечатанных исключениях. Это то же поведение, что и при нажатии на имя файла в журнале.
4. **Использовать мягкие переносы** https://developer.android.com/studio/images/buttons/logcat-wrap.png : нажмите, чтобы включить перенос строк и предотвратить горизонтальную прокрутку (хотя для любых неразрывных строк все равно потребуется горизонтальная прокрутка).
5. **Печать** https://developer.android.com/studio/images/buttons/logcat-print.png : Нажмите, чтобы распечатать сообщения LogCat. После выбора настроек печати в появившемся диалоговом окне вы также можете сохранить файл в формате PDF.
6. **Перезагрузка** https://developer.android.com/studio/images/buttons/logcat-restart.png : нажмите, чтобы очистить журнал и перезапустить logcat. В отличие от кнопки **Очистить logcat** , она восстанавливает и отображает предыдущие сообщения журнала, поэтому наиболее полезна, если Logcat перестает отвечать на запросы и вы не хотите терять сообщения журнала.
7. **Заголовок LogCat** https://developer.android.com/studio/images/buttons/logcat-headers.png : Нажмите, чтобы открыть диалоговое окно « **Настроить заголовок LogCat** », где вы можете настроить внешний вид каждого сообщения LogCat, например, показывать ли дату и время.
8. [Снимок](https://developer.android.com/studio/debug/am-screenshot.html)**экрана**https://developer.android.com/studio/images/buttons/monitor-screenshot.png : нажмите, чтобы [сделать снимок экрана](https://developer.android.com/studio/debug/am-screenshot.html) .
9. **Запись экрана** https://developer.android.com/studio/images/buttons/logcat-record.png : нажмите, чтобы [записать видео](https://developer.android.com/studio/debug/am-video.html) устройства (не более 3 минут).

Написать сообщения журнала

[Log](https://developer.android.com/reference/android/util/Log.html)Класс позволяет создавать лог - сообщения , которые появляются в LogCat. Как правило, вы должны использовать следующие методы журнала, перечисленные в порядке от наивысшего к низшему приоритету (или, от наименьшего к большинству подробного):

* [Log.e(String, String)](https://developer.android.com/reference/android/util/Log.html#e(java.lang.String,%20java.lang.String)) (ошибка)
* [Log.w(String, String)](https://developer.android.com/reference/android/util/Log.html#w(java.lang.String,%20java.lang.String)) (предупреждение)
* [Log.i(String, String)](https://developer.android.com/reference/android/util/Log.html#i(java.lang.String,%20java.lang.String)) (Информация)
* [Log.d(String, String)](https://developer.android.com/reference/android/util/Log.html#d(java.lang.String,%20java.lang.String)) (Отладка)
* [Log.v(String, String)](https://developer.android.com/reference/android/util/Log.html#v(java.lang.String,%20java.lang.String)) (подробный)

Смотрите [Log](https://developer.android.com/reference/android/util/Log.html)описание класса для более полного списка опций.

Вы никогда не должны компилировать подробные журналы в свое приложение, кроме как во время разработки. Журналы отладки компилируются, но удаляются во время выполнения, а журналы ошибок, предупреждений и информации всегда сохраняются.

Для каждого метода журнала первый параметр должен быть уникальным тегом, а второй параметр - сообщением. Тег сообщения системного журнала представляет собой короткую строку, указывающую системный компонент, из которого исходит сообщение (например, [ActivityManager](https://developer.android.com/reference/android/app/ActivityManager.html)). Ваш тег может быть любой строкой, которая будет вам полезна, например, именем текущего класса.

Хорошим соглашением является объявление TAGконстанты в вашем классе для использования в первом параметре. Например, вы можете создать сообщение журнала информации следующим образом:

[КОТЛИН](https://developer.android.com/studio/debug/am-logcat#kotlin)[ДЖАВА](https://developer.android.com/studio/debug/am-logcat#java)

private const val TAG = "MyActivity" ... Журнал . i ( TAG , «MyClass.getView () - получить номер позиции $ position» )

**Примечание.** Имена тегов длиной более 23 символов усекаются в выводе logcat.

Формат сообщения Logcat

Каждое сообщение журнала Android имеет тег и приоритет, связанный с ним. Тег сообщения системного журнала представляет собой короткую строку, указывающую системный компонент, из которого исходит сообщение (например, ActivityManager). Определяемым пользователем тегом может быть любая строка, которая будет вам полезна, например имя текущего класса (рекомендуемый тег). Вы определяете это в Log вызове метода, например:

[КОТЛИН](https://developer.android.com/studio/debug/am-logcat#kotlin)[ДЖАВА](https://developer.android.com/studio/debug/am-logcat#java)

Войти . d ( тег , сообщение )

Приоритет является одним из следующих значений:

* **V:** подробный (самый низкий приоритет)
* **D:** отладка
* **Я:** Информация
* **W:** Предупреждение
* **E:** ошибка
* **A:** Утверждать

Формат сообщения журнала:

***дата и*** ***время*** ***PID*** *-* ***TID*** */* ***приоритет пакета*** */* ***тег*** *:* ***сообщение***

Например, следующее сообщение журнала имеет приоритет Vи тег AuthZen:

12-10 13: 02: 50.071 1901-4229 / com.google.android.gms V / AuthZen: обработка намерений делегата.

PID обозначает идентификатор процесса, а TID - идентификатор потока; они могут быть одинаковыми, если есть только одна нить.

Установите уровень журнала

Вы можете контролировать, сколько сообщений появляется в logcat, установив уровень журнала. Вы можете отобразить все сообщения или только сообщения, указывающие на самые суровые условия.

Помните, что logcat продолжает собирать все сообщения независимо от настроек уровня журнала. Настройка просто определяет, что отображает logcat.

В меню Уровень журнала выберите одно из следующих значений:

* **Подробно:** Показать все сообщения журнала (по умолчанию).
* **Отладка:** Показать сообщения журнала отладки, которые полезны только во время разработки, а также уровни сообщений ниже в этом списке.
* **Информация:** Показать ожидаемые сообщения журнала для регулярного использования, а также уровни сообщений ниже в этом списке.
* **Предупреждать:** Показать возможные проблемы, которые еще не являются ошибками, а также уровни сообщений ниже в этом списке.
* **Ошибка:** Показать проблемы, которые вызвали ошибки, а также уровень сообщений ниже в этом списке.
* **Утверждение:** Показать проблемы, которые разработчик ожидает, никогда не должно случиться.

Поиск сообщений logcat

Для поиска сообщений, отображаемых в данный момент в logcat:

1. При желании выберите **Regex,** если вы хотите использовать шаблон поиска регулярного выражения.
2. Введите последовательность символов в поле поиска https://developer.android.com/images/tools/am-isearch.png.

Дисплей вывода logcat изменяется соответственно.

1. Нажмите Enter, чтобы сохранить строку поиска в меню во время этого сеанса.
2. Чтобы повторить поиск, выберите его в меню поиска. Выберите или отмените выбор **Regex по** мере необходимости (настройка не запоминается).

Фильтровать сообщения logcat

Один из способов уменьшить вывод журнала до управляемого уровня - ограничить его с помощью фильтра.

**Примечание** . Фильтр применяется к вашей полной истории logcat, а не только к тем сообщениям, которые в данный момент отображаются в logcat. Убедитесь, что другие параметры отображения установлены правильно, чтобы вы могли видеть выходные данные фильтра, которые вы хотите проверить.

Чтобы определить и применить фильтр:

1. В меню фильтра выберите параметр фильтра:
   * **Показать только выбранное приложение:** Показать сообщения, созданные только кодом приложения (по умолчанию). Logcat фильтрует сообщения журнала, используя PID активного приложения.
   * **Нет фильтров: не** применять фильтры. Logcat отображает все сообщения журнала с устройства, независимо от того, какой процесс вы выбрали.
   * **Изменить конфигурацию фильтра:** создать или изменить пользовательский фильтр. Например, вы можете создать фильтр для просмотра сообщений журнала из двух приложений одновременно.

После того, как вы определите фильтры, вы также можете выбрать их в меню. Чтобы удалить их из меню, удалите их.

1. Если вы выбрали **Изменить конфигурацию фильтра** , создайте или измените фильтр:
   * Укажите параметры *фильтра в* диалоговом окне « *Создать новый фильтр Logcat* »:
     + **Имя фильтра:** введите имя фильтра, который вы хотите определить, или выберите его на левой панели, чтобы изменить существующий фильтр. Имя может содержать только строчные буквы, символы подчеркивания и цифры.
     + **Метка журнала:** необязательно укажите тег. Для получения дополнительной информации см. [Формат сообщения logcat](https://developer.android.com/studio/debug/am-logcat#format) .
     + **Сообщение журнала: при** желании укажите текст сообщения журнала. Для получения дополнительной информации см. [Формат сообщения logcat](https://developer.android.com/studio/debug/am-logcat#format) .
     + **Имя пакета: при** желании укажите имя пакета. Для получения дополнительной информации см. [Формат сообщения logcat](https://developer.android.com/studio/debug/am-logcat#format) .
     + **PID: при** желании укажите идентификатор процесса. Для получения дополнительной информации см. [Формат сообщения logcat](https://developer.android.com/studio/debug/am-logcat#format) .
     + **Уровень журнала:** При желании выберите уровень журнала. Для получения дополнительной информации см. [Установка уровня журнала](https://developer.android.com/studio/debug/am-logcat#level) .
     + **Regex:** выберите этот параметр, чтобы использовать синтаксис регулярного выражения для этого параметра.
   * Нажмите **+,** чтобы добавить определение фильтра на левую панель.

Чтобы удалить фильтр, выберите его на левой панели и нажмите **-** .

* + Когда вы закончите, нажмите **ОК** .

Если вы не думаете, что видите нужные сообщения журнала, попробуйте выбрать « **Нет фильтров»** и выполнить [поиск](https://developer.android.com/studio/debug/am-logcat#searching) определенных сообщений журнала.

Читать сообщения сборки мусора

Иногда, когда происходит событие сборки мусора, они печатаются в logcat.

Для получения более подробной информации о памяти вашего приложения используйте [Memory Profiler](https://developer.android.com/studio/profile/memory-profiler.html) .

Журнал сообщений Dalvik

В Dalvik (но не в ART) каждый GC печатает следующую информацию в logcat:

D / dalvikvm ( ***PID*** ): ***GC\_Reason***  ***Amount\_freed*** , ***Heap\_stats*** , ***External\_memory\_stats*** , ***Pause\_time***

Пример:

D / dalvikvm (9050): GC\_CONCURRENT освобожден 2049K, 65% свободен 3571K / 9991K, внешний 4703K / 5261K, приостановлен 2 мс + 2 мс

**GC Reason**

Что вызвало GC и что это за коллекция. Возможные причины включают в себя:

GC\_CONCURRENT

Параллельный GC, который освобождает память, когда ваша куча начинает заполняться.

GC\_FOR\_MALLOC

GC был вызван, потому что ваше приложение пыталось выделить память, когда ваша куча уже была заполнена, поэтому система должна была остановить ваше приложение и восстановить память.

GC\_HPROF\_DUMP\_HEAP

GC, который появляется, когда вы запрашиваете создание файла HPROF для анализа вашей кучи.

GC\_EXPLICIT

Явный GC, например, когда вы звоните [gc()](https://developer.android.com/reference/java/lang/System.html" \l "gc())(что следует избегать вызова и вместо этого доверять GC при необходимости).

GC\_EXTERNAL\_ALLOC

Это происходит только на уровне API 10 и ниже (более новые версии выделяют все в куче Dalvik). GC для внешней распределенной памяти (такой как данные пикселей, хранящиеся в собственной памяти или байтовые буферы NIO).

**Освобожденная сумма**

Объем памяти, восстановленный из этого GC.

**Куча статистики**

Процент, свободный от кучи и (количество живых объектов) / (общий размер кучи).

**Статистика внешней памяти**

Внешне выделенная память на уровне API 10 и ниже (объем выделенной памяти) / (предел, при котором будет происходить сбор).

**Время паузы**

Большие кучи будут иметь большее время паузы. Одновременные паузы показывают две паузы: одну в начале коллекции и другую в конце.

Пока эти сообщения журнала накапливаются, обратите внимание на увеличение статистики кучи ( 3571K/9991Kзначение в приведенном выше примере). Если это значение продолжает увеличиваться, возможно, имеется утечка памяти.

Сообщения журнала ART

В отличие от Dalvik, ART не регистрирует сообщения для GC, которые не были явно запрошены. ГХ печатаются только тогда, когда они считаются медленными. Точнее, если пауза ГХ превышает 5 мс или длительность ГХ превышает 100 мс. Если приложение не находится в состоянии восприятия паузы (например, когда приложение находится в фоновом режиме, где пользователь не может получить паузу GC), то ни один из его GC не считается медленным. Явные GC всегда регистрируются.

ART включает в свои сообщения журнала сбора мусора следующую информацию:

I / art: ***GC\_Reason***  ***GC\_Name***  ***Objects\_freed*** ( ***Size\_freed*** ) AllocSpace Objects,

***Large\_objects\_freed*** ( ***Large\_object\_size\_freed*** ) ***Heap\_stats*** LOS объекты, ***Pause\_time (s)***

Пример:

I / art: явная одновременная развертка меток GC освобождает 104710 (7MB) объектов AllocSpace,

21 (416 КБ) LOS-объектов, 33% свободного, 25 МБ / 38 МБ, приостановлено 1.230 мс, всего 67.216 мс

**GC Reason**

Что вызвало GC и что это за коллекция. Возможные причины включают в себя:

Concurrent

Параллельный сборщик мусора, который не приостанавливает потоки приложения. Этот GC работает в фоновом потоке и не предотвращает выделения.

Alloc

GC был инициирован, потому что ваше приложение пыталось выделить память, когда ваша куча была уже заполнена. В этом случае сборка мусора произошла в выделяющем потоке.

Explicit

Сборка мусора была явно запрошена приложением, например, с помощью вызова [gc()](https://developer.android.com/reference/java/lang/System.html" \l "gc())или [gc()](https://developer.android.com/reference/java/lang/Runtime.html" \l "gc()). Как и Dalvik, в ART лучшая практика заключается в том, что вы доверяете GC и, по возможности, избегаете запрашивать явные GC. Явные GC не рекомендуется, потому что они блокируют поток распределения и излишне тратят циклы ЦП. Явные сборщики мусора могут также вызвать джанк (заикание, дрожание или остановку в приложении), если они вызывают прерывание других потоков.

NativeAlloc

Коллекция была вызвана внутренним давлением памяти от собственных выделений, таких как объекты размещения Bitmap или RenderScript.

CollectorTransition

Коллекция была вызвана переходом кучи; это вызвано изменением стратегии GC во время выполнения (например, когда приложение переключается между воспринимаемыми состояниями паузы). Переходы коллектора состоят из копирования всех объектов из свободного пространства свободного списка в выбранное пространство указателя (или наоборот).

Это происходит только на устройстве с низкой оперативной памятью до Android 8.0, когда приложение изменяет состояния процесса с состояния, воспринимаемого паузой (например, когда приложение находится на переднем плане, где пользователь может воспринимать паузу GC), на состояние, не воспринимаемое паузой (или наоборот).

HomogeneousSpaceCompact

Однородное сжатие пространства - это сжатие пространства свободного пространства до сжатия пространства свободного списка, которое обычно происходит, когда приложение переводится в незаметное для паузы состояние процесса. Основными причинами этого являются уменьшение использования оперативной памяти и дефрагментация кучи.

DisableMovingGc

Это не настоящая причина GC, но обратите внимание, что коллекция была заблокирована из-за использования GetPrimitiveArrayCritical. в то время как происходит одновременное сжатие кучи. В общем, использование GetPrimitiveArrayCritical настоятельно не рекомендуется из-за его ограничений на перемещение коллекторов.

HeapTrim

Это не причина GC, а примечание, что сбор был заблокирован до завершения обрезки кучи.

**Имя ГК**

ART имеет различные GC, которые можно запустить.

Concurrent mark sweep (CMS)

Целый сборщик кучи, который освобождает, собирает все пространства, кроме пространства изображения.

Concurrent partial mark sweep

В основном весь сборщик кучи, который собирает все пробелы, кроме изображения и зиготы.

Concurrent sticky mark sweep

Коллектор поколений, который может освобождать только объекты, выделенные с момента последнего сбора данных. Эта сборка мусора запускается чаще, чем полная или частичная разметка, поскольку она быстрее и имеет меньшие паузы.

Marksweep + semispace

Непараллельное копирование GC, используемое для переходов кучи, а также для однородного сжатия пространства (для дефрагментации кучи).

**Объекты освобождены**

Количество объектов, которые были восстановлены из этого GC из небольшого пространства объектов.

**Размер освобожден**

Количество байтов, которые были извлечены из этого GC из небольшого пространства объектов.

**Большие объекты освобождены**

Номер объекта в большом объектном пространстве, который был восстановлен из этой сборки мусора.

**Большой размер объекта освобожден**

Количество байтов в большом объектном пространстве, которые были восстановлены из этой сборки мусора.

**Куча статистики**

Процент свободных и (количество живых объектов) / (общий размер кучи).

**Время паузы**

В общем время паузы пропорционально количеству ссылок на объекты, которые были изменены во время работы GC. В настоящее время GC ART CMS имеет только одну паузу, ближе к концу GC. Движущиеся GC имеют длинную паузу, которая длится большую часть продолжительности GC.

Если вы видите большое количество GC в logcat, посмотрите на увеличение статистики кучи ( 25MB/38MBзначение в приведенном выше примере). Если это значение продолжает увеличиваться и кажется, что оно никогда не уменьшается, может возникнуть утечка памяти. В качестве альтернативы, если вы видите GC, который по причине «Alloc», то вы уже работаете около своей емкости кучи и можете ожидать исключения OOM в ближайшем будущем.