

## Уровни логирования и что они означают

Для начала разберёмся с логами. Это текстовые файлы, в которых записываются все действия пользователя. Например, какие кнопки он нажимает в приложении и как на это оно реагирует в ответ.

Записи в логах формируются в хронологическом порядке. Самая свежая — внизу.

Есть два вида логов:

- **Crash logs** — файл, в котором хранятся записи только об ошибках экстренного завершения программы — по-простому, когда приложение крашнулось.
- **Logs** — простые логи, или журнал событий. Это файл, в котором хранятся системные записи и ответы устройства на действие пользователя.

Логи на мобильных устройствах бывают нескольких уровней:

- **ERROR,**
- **WARN,**
- **INFO,**
- **DEBUG,**
- **VERBOSE.**

Они представлены по уровню важности — от самого высокого к самому низкому, — и каждый следующий уровень включает в себя предыдущий. Например, VERBOSE содержит в себе логи всех остальных.

### Error (ERROR)

На этом уровне информируются ошибки работы системы.

Записи этого уровня требуют быстрого вмешательства разработчика — на такие ошибки нужно реагировать максимально быстро.

Как пример, такая запись в логе:

*“ [ZeroHung]zrhung\_get\_config: Get config failed for wp[0x0008] ] ”*

Эта системная ошибка сообщает, что происходит утечка памяти при взаимодействии с каким-то элементом или приложением.

### Warning (WARN)

На этом уровне отображаются записи, сообщающие о каком-то неожиданном поведении, требующем внимания, или о ситуации, которая незнакома системе.

Например, сообщение ниже — запись из тестового приложения:

*“ [OMX.hisi.video.decoder.avc] setting nBufferCountActual to 16 failed: -2147483648 “*

Мы пытаемся декодировать запись в какой-то формат, но его нет. Ошибка сообщает о неуспешной попытке настройки видеоплеера в нужном формате.

Ещё пример:

*“ BroadcastQueue: Permission Denial: broadcasting Intent ”*

Эта системная ошибка говорит о сбое в работе одного из виджетов на устройстве.

## **Info (INFO)**

На этот уровень приходят записи информационного характера, например о работе системы.

Допустим, такое сообщение об уровне заряда батареи на устройстве:

*“ APwBatteryMonitor: screen off start battery: 100 ”*

А это сообщение говорит о том, что экран устройства был выключен:

*“ HwBatteryService: intent = Intent { act=android.intent.action.SCREEN\_OFF  
flag=0x58200010 } ”*

## **Debug (DEBUG)**

Это уровень сообщений, в которых передаётся информация о процессах отладки или шагах работы крупных процессов.

Например, в записи ниже сказано, что пользователь нажимал на кнопку уменьшения или увеличения громкости:

*“ MediaSessionService: dispatchVolumeKeyEvent ”*

Сначала мы видим запись о самом факте нажатия на кнопку, далее оно расшифровывается подробнее:

*{ action=ACTION\_DOWN, keyCode=KEYCODE\_VOLUME\_UP }*

Ещё пример: если ваше приложение использует сокет-сессию, то на уровне DEBUG мы можем увидеть, когда сессия начинается и заканчивается:

*“ b\$b: WebSocket connected ”*

## **Verbose (VERBOSE)**

Сообщения такого уровня уточняют или раскрывают действия.

Например, у нас есть служба управления окнами на экране приложения. И на уровне Verbose мы можем увидеть подробности её работы.

Открытие окна:

*WindowManager: addWindow*

Закрытие окна:

*WindowManager: Removing Window*

На этом уровне мы можем посмотреть системные подробности наших действий. Например, при включении геолокации в записи отобразится текущая геолокация.

*GnssLocationProvider: reportLocation Location [...]*

А меняя звук на устройстве, мы увидим, как растёт или падает значение:

*AudioManager: getStreamVolume streamType: 3 volume: 10*

Каждое нажатие, то есть изменение звука, будет отражаться новым сообщением.

Verbose — уровень самого низкого приоритета. Выбирая такой уровень отображения логов, мы будем видеть записи и со всех предыдущих уровней.

## Инструменты для снятия логов: Android

1. **Logcat** в составе **Android Studio**, самый известный и широко используемый.
2. Выгрузка логов с самого устройства. Кроме режима разработчика нам нужно подключить устройство к ПК через USB и установить ADB — **Android Debug Bridge**.
3. **SDK Platform Tools**

## Инструменты снятия логов: iOS

1. **xCode** — интегрированная среда разработки (IDE), в которую встроен инструмент **Simulator**.
2. **iMazing** поможет снимать iOS-логи для тех, у кого установлен Windows.

1. В первом столбце — дата и время поступления записи.
2. Во втором — обозначения уровней логирования. Например, D — это Debug.

- # ios

1 — дата и время получения сообщения; 2 — имя телефона, информация, с какой части устройства пришло сообщение, и описание; 3 — поисковая строка для фильтрации выдачи