Cómo escribir y ver registros con Logcat

La ventana **Logcat** de Android Studio muestra los mensajes del sistema, por ejemplo, cuando se produce la recolección de elementos no utilizados, y los mensajes que agregas a tu app con la clase [Log](https://developer.android.com/reference/android/util/Log.html). Se muestran los mensajes en tiempo real y se conserva un historial para que puedas ver los mensajes más antiguos.

A fin de mostrar solo la información de interés, puedes crear filtros, modificar la cantidad de información que se muestra en los mensajes, establecer niveles de prioridad, mostrar los mensajes generados únicamente por el código de la app y realizar búsquedas en el registro. De forma predeterminada, logcat solo muestra el resultado del registro relacionado con la última app que se ejecutó.

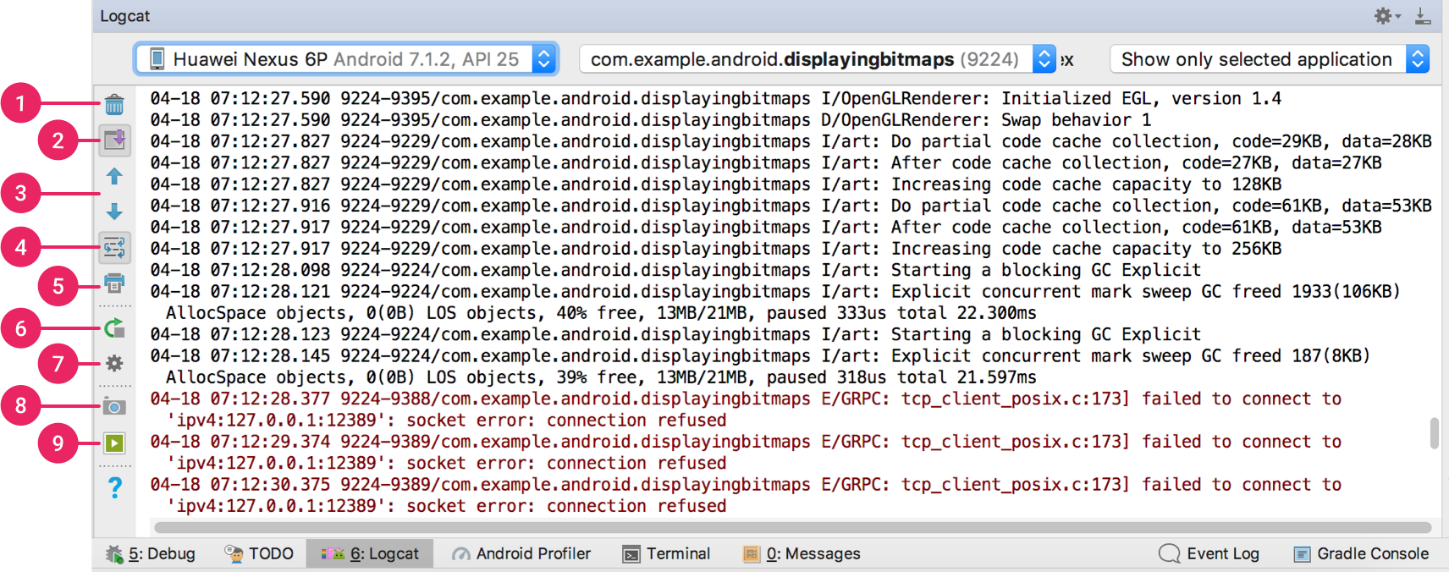
Si la app genera una excepción, logcat muestra un mensaje y, a continuación, el seguimiento de pila que contiene los vínculos a la línea de código.

A partir de Android Studio 2.2, también se muestran los mensajes del registro para la app en ejecución en la ventana **Run**. Ten en cuenta que puedes configurar la pantalla de resultados de logcat, pero no la ventana Run.

Cómo ver los registros de tu app

Para ver los mensajes del registro de una app, haz lo siguiente:

1. [Compila y ejecuta la app](https://developer.android.com/studio/run/index.html#RunningApp) en un dispositivo.
2. Haz clic en **View > Tool Windows > Logcat** (o en Logcat  en la barra de la ventana de herramientas).

En la ventana "Logcat", se muestran los mensajes del registro de la app que seleccionas en las listas desplegables de la parte superior de la ventana, como se muestra en la figura 1.

**Figura 1:** Ventana "Logcat"

De forma predeterminada, logcat muestra solo los mensajes del registro de la app que se está ejecutando en el dispositivo. Para cambiar este valor predeterminado, consulta cómo [filtrar los mensajes de logcat](https://developer.android.com/studio/debug/am-logcat.html#filtering).

En la barra de herramientas de Logcat, se incluyen los siguientes botones:

1. **Clear logcat** : Haz clic en esta opción para borrar el registro visible.
2. **Scroll to the end** : Haz clic para saltar a la parte inferior del registro y ver los últimos mensajes. Si, a continuación, haces clic en una línea del registro, se detendrá el desplazamiento de la vista en ese punto.
3. **Up the stack trace**  y Down the stack trace : Haz clic para navegar hacia arriba y hacia abajo por los seguimientos de pila del registro, y selecciona los nombres de archivo subsiguientes que se muestran en las excepciones generadas (y observa los números de línea correspondientes en el editor). Este comportamiento es el mismo que se genera cuando haces clic en un nombre de archivo en el registro.
4. **Use soft wraps** : Haz clic para habilitar el ajuste de línea y evitar el desplazamiento horizontal (aunque las strings no separables requerirán el desplazamiento horizontal).
5. **Print** : Haz clic para imprimir los mensajes de logcat y selecciona las preferencias de impresión en el diálogo que se muestra; puedes optar también por guardar los mensajes en un PDF.
6. **Restart** : Haz clic para borrar el registro y reiniciar logcat. A diferencia del botón **Clear logcat**, este botón recupera y muestra los mensajes anteriores del registro, de modo que es muy útil si Logcat no responde y no quieres perder los mensajes.
7. **Logcat header** : Haz clic para abrir el diálogo Configure Logcat Header, donde puedes personalizar la apariencia de los mensajes de logcat, por ejemplo, si quieres que se muestre la fecha y la hora.
8. **Screen capture** : Haz clic para [tomar una captura de pantalla](https://developer.android.com/studio/debug/am-screenshot.html).
9. **Screen record** : Haz clic para [grabar un video](https://developer.android.com/studio/debug/am-video.html) del dispositivo (durante un máximo de 3 minutos).

Cómo escribir mensajes del registro

La clase [Log](https://developer.android.com/reference/android/util/Log.html) te permite crear mensajes del registro que se muestran en logcat. Por lo general, debes usar los siguientes métodos de registro, que se ordenan de mayor a menor prioridad (o del más detallado al menos detallado).

* [Log.e(String, String)](https://developer.android.com/reference/android/util/Log.html#e(java.lang.String,%20java.lang.String)) (error)
* [Log.w(String, String)](https://developer.android.com/reference/android/util/Log.html#w(java.lang.String,%20java.lang.String)) (advertencia)
* [Log.i(String, String)](https://developer.android.com/reference/android/util/Log.html#i(java.lang.String,%20java.lang.String)) (información)
* [Log.d(String, String)](https://developer.android.com/reference/android/util/Log.html#d(java.lang.String,%20java.lang.String)) (depuración)
* [Log.v(String, String)](https://developer.android.com/reference/android/util/Log.html#v(java.lang.String,%20java.lang.String)) (registro detallado)

Consulta la descripción de la clase [Log](https://developer.android.com/reference/android/util/Log.html) para obtener una lista más completa de opciones.

No debes compilar registros detallados en tu app, salvo durante el desarrollo. Los registros de depuración se compilan seccionados durante el tiempo de ejecución, y siempre se conservan los registros de error, advertencia e información.

Para cada método de registro, el primer parámetro debe ser una etiqueta exclusiva, mientras que el segundo es el mensaje. La etiqueta de un mensaje de registro del sistema es una string corta que indica el componente del sistema desde el que se origina el mensaje (por ejemplo, [ActivityManager](https://developer.android.com/reference/android/app/ActivityManager.html)). La etiqueta puede ser cualquier string que resulte útil, como el nombre de la clase actual.

Una buena convención es declarar una constante TAG en tu clase para usarla en el primer parámetro. Por ejemplo, puedes crear un mensaje de registro de información de la siguiente manera:

private static final String TAG = "MyActivity";  
    ...  
    Log.i(TAG, "MyClass.getView() — get item number " + position);

**Nota:** Los nombres de etiquetas con más de 23 caracteres aparecerán truncados en el resultado de logcat.

Formato de mensajes de logcat

Todos los mensajes de registro de Android tienen una etiqueta y una prioridad asociadas a ellos. La etiqueta de un mensaje de registro del sistema es una string corta que indica el componente del sistema desde el que se origina el mensaje (por ejemplo, ActivityManager). Una etiqueta definida por el usuario puede ser cualquier string que resulte útil, como el nombre de la clase actual (la etiqueta recomendada). Se define en una llamada al método Log, por ejemplo:

Log.d(tag, message);

La prioridad es uno de los siguientes valores:

* **V:** Registro detallado (prioridad más baja)
* **D:** Depuración
* **I:** Información
* **W:** Advertencia
* **E:** Error
* **A:** Aserción

El formato del mensaje de registro es el siguiente:

***date******time******PID****-****TID****/****package******priority****/****tag****:* ***message***

Por ejemplo, el siguiente mensaje de registro tiene la prioridad V y una etiqueta AuthZen:

12-10 13:02:50.071 1901-4229/com.google.android.gms V/AuthZen: Handling delegate intent.

"PID" es el identificador del proceso y "TID" es el identificador de subproceso, y pueden ser iguales si hay un único subproceso.

Cómo establecer el nivel de registro

Para controlar la cantidad de mensajes que se muestran en logcat, puedes establecer el nivel de registro. Puedes mostrar todos los mensajes o solo aquellos que indiquen las condiciones más graves.

Recuerda que logcat continúa recopilando todos los mensajes, independientemente del nivel de registro que establezcas. Este ajuste solo determina qué mostrará logcat.

En el menú de nivel de registro, selecciona uno de los siguientes valores:

* **Verbose:** Se muestran todos los mensajes del registro (configuración predeterminada).
* **Debug:** Se muestran los mensajes del registro que son útiles solo para la etapa de desarrollo, además de los niveles de mensajes de la parte inferior de la lista.
* **Info:** Se muestran los mensajes de registro esperados para uso normal, además de los niveles de mensajes de la parte inferior de la lista.
* **Warn:** Se muestran los posibles problemas que aún no han causado errores, además de los niveles de mensajes de la parte inferior de la lista.
* **Error:** Se muestran los problemas que han causado errores, además de los niveles de mensajes de la parte inferior de la lista.
* **Assert:** Se muestran los problemas que el desarrollador cree que no ocurrirán en ningún momento.

Cómo buscar mensajes en logcat

Para buscar los mensajes que se muestran actualmente en logcat:

1. De manera opcional, selecciona **Regex** si quieres usar un patrón de búsqueda de expresiones regulares.
2. Escribe una secuencia de caracteres en el campo de búsqueda .

La pantalla de resultados de logcat cambiará según corresponda.

1. Presiona Intro para almacenar la string de búsqueda en el menú durante esta sesión.
2. Para repetir una búsqueda, selecciónala en el menú de búsqueda. Selecciona **Regex** o anula su selección según sea necesario (no se recuerda esta configuración).

Cómo filtrar mensajes en logcat

Una forma de reducir los resultados del registro a un nivel razonable es restringirlos mediante un filtro.

**Nota:** El filtro se aplica a todo el historial de logcat, no solo a los mensajes que se muestran actualmente. Asegúrate de que las demás opciones de visualización estén configuradas de forma adecuada, de modo que puedas ver el resultado del filtro que quieras examinar.

Para definir y aplicar un filtro, sigue estos pasos:

1. En el menú de filtros, selecciona una opción de filtro:
   * **Show only selected application:** Muestra solo los mensajes generados por el código de la app (configuración predeterminada). Logcat filtra los mensajes del registro mediante el PID de la app activa.
   * **No Filters:** No se aplica ningún filtro. Logcat muestra todos los mensajes del registro del dispositivo, independientemente del proceso seleccionado.
   * **Edit Filter Configuration:** Permite crear o modificar un filtro personalizado. Por ejemplo, puedes crear un filtro para ver los mensajes del registro de dos apps al mismo tiempo.

Una vez definidos los filtros, también puedes seleccionarlos en el menú. Para quitarlos del menú, debes borrarlos.

1. Si seleccionaste **Edit Filter Configuration**, crea o modifica un filtro:
   * Especifica los parámetros del filtro en el diálogo *Create New Logcat Filter*:
     + **Filter Name:** Escribe el nombre del filtro que quieras definir o selecciónalo en el panel izquierdo si quieres modificar un filtro existente. El nombre solo puede contener caracteres en minúscula, guiones bajos y números.
     + **Log Tag:** Permite especificar una etiqueta de manera opcional. Para obtener más información, consulta [Formato de mensajes de logcat](https://developer.android.com/studio/debug/am-logcat.html#format).
     + **Log Message:** Permite especificar un texto de mensaje de registro de manera opcional. Para obtener más información, consulta [Formato de mensajes de logcat](https://developer.android.com/studio/debug/am-logcat.html#format).
     + **Package Name:** Permite especificar un nombre de paquete de manera opcional. Para obtener más información, consulta [Formato de mensajes de logcat](https://developer.android.com/studio/debug/am-logcat.html#format).
     + **PID:** Puedes especificar un ID de proceso de manera opcional. Para obtener más información, consulta [Formato de mensajes de logcat](https://developer.android.com/studio/debug/am-logcat.html#format).
     + **Log Level:** Permite seleccionar un nivel de registro de manera opcional. Para obtener más información, consulta [Cómo configurar el nivel de registro](https://developer.android.com/studio/debug/am-logcat.html#level).
     + **Regex:** Selecciona esta opción a fin de usar la sintaxis de expresiones regulares para el parámetro.
   * Haz clic en **+** para agregar la definición del filtro al panel izquierdo.

Para quitar un filtro, selecciónalo en el panel izquierdo y haz clic en **-**.

* + Cuando hayas terminado, haz clic en **OK**.

Si crees que no se están mostrando los mensajes del registro que quieres, intenta seleccionando **No filters** y [buscando](https://developer.android.com/studio/debug/am-logcat.html#searching) mensajes del registro específicos.

Cómo leer mensajes de recolección de elementos no utilizados

A veces, cuando se producen eventos de recolección de elementos no utilizados, se incluyen en logcat.

Para obtener más información acerca de la memoria de la app, usa [Memory Profiler](https://developer.android.com/studio/profile/memory-profiler.html).

Mensajes del registro de Dalvik

En Dalvik (pero no en ART), todas las recolecciones de elementos no utilizados envían la siguiente información a logcat:

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

Ejemplo:

D/dalvikvm( 9050): GC\_CONCURRENT freed 2049K, 65% free 3571K/9991K, external 4703K/5261K, paused 2ms+2ms

**GC Reason**

Indica qué activó la recolección de elementos no utilizados y qué tipo de recopilación es. Estos son algunos de los posibles motivos:

GC\_CONCURRENT

Es una recolección de elementos no utilizados simultánea que libera memoria cuando se inicia la pila para completarla.

GC\_FOR\_MALLOC

Se generó una recolección de elementos no utilizados porque la app intentó asignar memoria cuando la pila ya estaba completa, de modo que el sistema tuvo que detener la app y reclamar memoria.

GC\_HPROF\_DUMP\_HEAP

Se genera una recolección de elementos no utilizados cuando solicitas crear un archivo HPROF para analizar la pila.

GC\_EXPLICIT

Es una recolección de elementos no utilizados explícita, por ejemplo, cuando llamas a [gc()](https://developer.android.com/reference/java/lang/System.html#gc()) (debes evitar llamarlo y, en su lugar, debes permitir que se ejecute la recolección de elementos no utilizados cuando sea necesario).

GC\_EXTERNAL\_ALLOC

Esto sucede solo en la API nivel 10 y en versiones anteriores (las versiones más recientes asignan todo en la pila de Dalvik). Es una recolección de elementos no utilizados para memoria asignada de manera externa (por ejemplo, los datos de píxeles almacenados en la memoria nativa o en búferes de bytes NIO).

**Amount freed**

Es la cantidad de memoria reclamada desde esta recolección de elementos no utilizados.

**Heap stats**

Es el porcentaje libre del montón y (número de objetos en vivo)/(tamaño total del montón).

**External memory stats**

Es la memoria asignada de manera externa en la API nivel 10 y versiones anteriores (cantidad de memoria asignada)/(límite en el que se llevará a cabo la recopilación).

**Pause time**

Los montones más grandes tienen tiempos de pausa más largos. Los tiempos de pausa simultáneos muestran dos pausas: una al inicio de la recopilación y otra cerca del final.

Mientras se acumulan estos mensajes del registro, busca aumentos en las estadísticas del montón (el valor 3571K/9991K del ejemplo anterior). Si este valor continúa aumentando, es posible que haya una fuga de memoria.

Mensajes de registro de ART

A diferencia de Dalvik, ART no registra mensajes para las recolecciones de elementos no utilizados que no se solicitaron de forma explícita. Las recolecciones de elementos no utilizados solo se incluyen en el registro cuando se considera que son lentas. Para ser más precisos, cuando la recolección de elementos no utilizados supera los 5 ms o cuando la duración de recolección de elementos no utilizados supera los 100 ms. Si no se detuvo la app (por ejemplo, cuando está ejecutándose en segundo plano, donde el usuario no puede percibir la detención de la recolección de elementos no utilizados), entonces no se considera lenta ninguna de las recolecciones. Las recolecciones de elementos no utilizados explícitas siempre se registran.

ART incluye la siguiente información en los mensajes de registro de recolección de elementos no utilizados:

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

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

Ejemplo:

I/art : Explicit concurrent mark sweep GC freed 104710(7MB) AllocSpace objects,

21(416KB) LOS objects, 33% free, 25MB/38MB, paused 1.230ms total 67.216ms

**GC Reason**

Indica qué activó la recolección de elementos no utilizados y qué tipo de recopilación es. Estos son algunos de los posibles motivos:

Concurrent

Se trata de una recolección de elementos no utilizados que no suspende los subprocesos de la app. Esta recolección se ejecuta en un subproceso en segundo plano y no evita que se realicen asignaciones.

Alloc

Se inició la recolección porque la app intentó asignar memoria cuando el montón ya estaba completo. En este caso, la recolección tuvo lugar en el subproceso de asignación.

Explicit

Una app solicitó la recolección de elementos no utilizados, por ejemplo, mediante la llamada a [gc()](https://developer.android.com/reference/java/lang/System.html#gc()) o [gc()](https://developer.android.com/reference/java/lang/Runtime.html#gc()). Al igual que en Dalvik, en ART se recomienda confiar en la recolección y, en lo posible, evitar la solicitud de recolecciones explícitas. No se aconseja usar recolecciones de elementos no utilizados, ya que bloquean el subproceso de asignación y se desperdician ciclos de la CPU de manera innecesaria. También pueden ocasionar bloqueos (interrupción de flujo, inestabilidad o detenciones de la app) si interrumpen la activación de otros subprocesos.

NativeAlloc

Se activó la recolección como consecuencia de la presión de la memoria nativa de las asignaciones nativas; por ejemplo, objetos de asignación RenderScript o mapas de bits.

CollectorTransition

Se activó la recolección como consecuencia de una transición de montón, y esto se debe al cambio de la estrategia de recolección durante el tiempo de ejecución (por ejemplo, cuando la app cambia entre estados perceptibles de detención). Las transiciones del recolector consisten en copiar todos los objetos de un espacio de copia de seguridad de listas libre a un espacio de puntero separado (o viceversa).

Esto ocurre solo en dispositivos con poca RAM anteriores a Android 8.0 cuando una app cambia los estados del proceso, desde un estado perceptible de detención (por ejemplo, cuando se ejecuta la app en segundo plano, donde el usuario puede percibir un estado de detención de la recolección de elementos no utilizables) hasta un estado perceptible de no detención (o viceversa).

HomogeneousSpaceCompact

La compactación espacial homogénea consiste en la compactación de un espacio de listas libre que, por lo general, ocurre cuando una app pasa a un estado imperceptible de proceso de detención. Los motivos principales por los que se realiza esta acción son para reducir el uso de la RAM y desfragmentar el montón.

DisableMovingGc

Esto no constituye un motivo real de recolección de elementos no utilizados, sino una nota que indica que se bloqueó la recopilación debido al uso de GetPrimitiveArrayCritical durante la compactación simultánea del montón. En general, no se aconseja usar GetPrimitiveArrayCritical debido a las restricciones que plantea para el movimiento de los recolectores.

HeapTrim

Esto no constituye un motivo real de recolección de elementos no utilizados, sino una nota que indica que se bloqueó la recopilación hasta que finalizó el corte del montón.

**GC Name**

ART tiene varias recopilaciones de elementos no utilizados que se pueden ejecutar.

Concurrent mark sweep (CMS)

Es un recolector de montón completo que realiza recolecciones en todos los espacios, excepto los de imágenes.

Concurrent partial mark sweep

Es un recolector de montón completo que realiza recopilaciones en todos los espacios, excepto en los de imágenes y zygote.

Concurrent sticky mark sweep

Es un recolector generacional que solo puede liberar los objetos asignados después de la última recolección. Esta recolección de elementos no utilizados se ejecuta con mayor frecuencia que un esquema de “marcar y limpiar” total o parcial, debido a que es más rápida y tiene pausas menos prolongadas.

Marksweep + semispace

Es una recolección de copia no simultánea que se usa para transiciones de montón y compactación de espacios homogéneos (para desfragmentar el montón).

**Objects freed**

Es el número de objetos reclamados desde esta recolección del espacio de objetos pequeños.

**Size freed**

Es el número de bytes reclamados desde esta recolección del espacio de objetos pequeños.

**Large objects freed**

Es el número de objetos en el espacio de objetos grandes que se reclamaron desde esta recolección.

**Large object size freed**

Es el número de bytes en el espacio de objetos grandes que se reclamaron desde esta recolección.

**Heap stats**

Porcentaje libre y (cantidad de objetos activos)/(tamaño total del montón).

**Pause times**

En general, los tiempos de detención son proporcionales al número de referencias de objetos que se modificaron mientras se ejecutaba la recolección. Actualmente, las recolecciones de CMS de ART solo tienen una detención, cerca del final de la recolección. Las recolecciones en movimiento tienen una detención larga, que dura la mayor parte del tiempo de la recolección.

Si observas una gran cantidad de recolecciones en logcat, busca aumentos en las estadísticas del montón (el valor 25MB/38MB del ejemplo anterior). Si este valor continúa aumentando y no parece disminuir, es posible que haya una fuga de memoria. De manera alternativa, si ves una recolección que se debe al motivo "Alloc", entonces ya estás cerca de alcanzar la capacidad del montón y puedes esperar excepciones OOM en el futuro cercano.