Android Public Tracker 36967392

# SoundPoolThread causing SIGSEGV via JNI ERROR “accessed deleted global reference”

+1 7 Hotlists (3) Mark as Duplicate

Comments (7) Dependencies Duplicates (0) Blocking (0) Resources (2)

Obsolete Bug P3 + Add Hotlist [AOSP] assigned

STATUS UPDATE No update yet. Edit

DESCRIPTION ch...@gmail.com created issue #1

As detailed here:
http://stackoverflow.com/questions/13436470/soundpoolthread-causing-sigsegv-via-jni-error-accessed-deleted-global-reference

In a loop creating multiple SoundPool objects, and calling load on each, it is possible for a SoundPool java object to get destroyed before the asynchronous load completes and SoundPool::notify

I will attach the full logcat output. The error and backtrace are:

```
D/dalvikvm(27638): GC_FOR_ALLOC freed 1635K, 29% free 10029K/14076K, paused 15ms, total 15ms
I/dalvikvm-heap(27638): Grow heap (frag case) to 11.155MB for 1400848-byte allocation
D/dalvikvm(27638): GC_FOR_ALLOC freed <1K, 20% free 11397K/14076K, paused 15ms, total 15ms
I/dalvikvm(27638): JNI ERROR (app bug): accessed deleted global reference 0x1da00292
E/dalvikvm(27638): VM aborting
F/libc    (27638): Fatal signal 11 (SIGSEGV) at 0xdeadd00d (code=1), thread 28043 (SoundPoolThread)
I/DEBUG   (  156): *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
I/DEBUG   (  156): Build fingerprint: 'google/occam/mako:4.2.2/JDQ39/573038:user/release-keys'
I/DEBUG   (  156): Revision: '11'
I/DEBUG   (  156): pid: 27638, tid: 28043, name: SoundPoolThread  >>> com.flexpansion.android <<<
I/DEBUG   (  156): signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr deadd00d
I/DEBUG   (  156):     r0 00000000  r1 00000000  r2 deadd00d  r3 00000000
I/DEBUG   (  156):     r4 40e5a1b0  r5 0000020c  r6 1da00292  r7 771addf0
I/DEBUG   (  156):     r8 7064bfb0  r9 6fc77a63  sl 771addc0  fp 00000001
I/DEBUG   (  156):     ip 00004000  sp 771adb38  lr 40152109  pc 40df1c90  cpsr 60000030
I/DEBUG   (  156):     d0  74726f6261204d56  d1  657373656363616e
I/DEBUG   (  156):     d2  004d0049002e0075  d3  0072006f006d006c
I/DEBUG   (  156):     d4  002e00640069006f  d5  006900740061004c
I/DEBUG   (  156):     d6  00790065004b006e  d7  00720061006f0062
I/DEBUG   (  156):     d8  4216584000000077  d9  4216584041fe3c88
I/DEBUG   (  156):     d10 00000000428a0000  d11 0000000000000000
I/DEBUG   (  156):     d12 0000000000000000  d13 0000000000000000
I/DEBUG   (  156):     d14 0000000000000000  d15 0000000000000000
I/DEBUG   (  156):     d16 0000000800000007  d17 7e37e43c8800759c
I/DEBUG   (  156):     d18 0000000000000000  d19 3ff2b851eb851eb8
I/DEBUG   (  156):     d20 7132211071322110  d21 12080503632b1d0e
I/DEBUG   (  156):     d22 0707070703030303  d23 0000000000000000
I/DEBUG   (  156):     d24 008f008f008f008f  d25 008f008f008f008f
I/DEBUG   (  156):     d26 009d009d009d009d  d27 00ee00ee00ee00ee
I/DEBUG   (  156):     d28 0100010001000100  d29 0100010001000100
I/DEBUG   (  156):     d30 0003000000030000  d31 0003000000030000
I/DEBUG   (  156):     scr 60000090
I/DEBUG   (  156):
I/DEBUG   (  156): backtrace:
I/DEBUG   (  156):     #00  pc 00045c90  /system/lib/libdvm.so (dvmAbort+75)
I/DEBUG   (  156):     #01  pc 000285ec  /system/lib/libdvm.so (IndirectRefTable::get(void*) const+336)
I/DEBUG   (  156):     #02  pc 0004a225  /system/lib/libdvm.so (dvmDecodeIndirectRef(Thread*, _jobject*)+80)
I/DEBUG   (  156):     #03  pc 0005fbb7  /system/lib/libdvm.so (dvmCallMethodV(Thread*, Method const*, Object*, bool, JValue*, std::__va_list)+150)
I/DEBUG   (  156):     #04  pc 000499fb  /system/lib/libdvm.so
I/DEBUG   (  156):     #05  pc 00000ee3  /system/lib/libsoundpool.so
I/DEBUG   (  156):     #06  pc 00000f17  /system/lib/libsoundpool.so
I/DEBUG   (  156):     #07  pc 00058ac7  /system/lib/libmedia.so (android::SoundPool::notify(android::SoundPoolEvent)+46)
I/DEBUG   (  156):     #08  pc 00059d93  /system/lib/libmedia.so (android::SoundPoolThread::doLoadSample(int)+92)
I/DEBUG   (  156):     #09  pc 00059ddd  /system/lib/libmedia.so (android::SoundPoolThread::run()+56)
I/DEBUG   (  156):     #10  pc 0004679f  /system/lib/libandroid_runtime.so (android::AndroidRuntime::javaThreadShell(void*)+66)
I/DEBUG   (  156):     #11  pc 00010dcd  /system/lib/libutils.so
I/DEBUG   (  156):     #12  pc 0000e3d8  /system/lib/libc.so (__thread_entry+72)
I/DEBUG   (  156):     #13  pc 0000dac4  /system/lib/libc.so (pthread_create+160)
```

```
void SoundPool::notify(SoundPoolEvent event)
{
   Mutex::Autolock lock(&mCallbackLock);
   if (mCallback != NULL) {
      mCallback(event, this, mUserData);
   }
}
```

deleted

0 B ⓘ

✓ Links (2)

“http://stackoverflow.com/questions/13436470/soundpoolthread-causing-sigsegv-via-jni-error-accesse… ”

“ …our best to address the issue reported, however our product team has shifted work priority which doesn't include this issue. For now, we will be closing the issue as "Won't Fix (Obsolete)". If this issu

COMMENTS

**ch...@gmail.com** <ch...@gmail.com> #2

This happens in Android 4.2.2 on Nexus 4.

**fa...@gmail.com** <fa...@gmail.com> #3

Confirmed on Nexus 7 (Android 4.2).

**fa...@gmail.com** <fa...@gmail.com> #4

Actually, in my scenario (Nexus 7), it wasn't multiple *SoundPool** object, but multiple sound effect loading with in a *single* SoundPool. This is about what my stackoverflow topic is about.

**ch...@gmail.com** <ch...@gmail.com> #5

It's almost certainly the same issue. I just happened to observe it occurring in a loop where each iteration created a single SoundPool object. The root cause appears to be the Java object be has just run. Looping over the alloc code is not a cause, it just means we are more likely to hit the problem, since it is timing dependent.

**en...@google.com** <en...@google.com> #6

*Assigned to gk...@google.com.*

gkasten: do you own SoundPool?

**gk...@android.com** <gk...@android.com>

*Reassigned to gk...@android.com.*

**sa...@google.com** <sa...@google.com> #7

*Status: Won't Fix (Obsolete)*

Thank you for your feedback. We have tried our best to address the issue reported, however our product team has shifted work priority which doesn't include this issue. For now, we will be clo