Deleted JNI references print spurious errors to logcat

Infeasible Bug P2 + Add Hotlist

STATUS UPDATE No update yet. Edit

DESCRIPTION em...@google.com created issue #1

Note: This is an issue originally reported by Epic Games.

Build: AI-203.7717.56.2031.7935034, 202111210535,

AI-203.7717.56.2031.7935034, JRE 11.0.10+0-b96-7249189x64 JetBrains s.r.o, OS Windows 10(amd64) v10.0 , screens 9600.0x5400.0

AS: Arctic Fox | 2020.3.1 Patch 4; Kotlin plugin: 203-1.5.20-release-289-AS7717.8; Android Gradle Plugin: 7.0.4; Gradle: 7.0.2; Gradle JDK: version 11.0.10; NDK: from local.properties: (not specifie found); LLDB: pinned revision 3.1 not found, latest from SDK: (package not found); CMake: from local.properties: (not specified), latest from SDK: 3.18.1-g262b901-dirty, from PATH: (not found)

Repro instructions:

1. Create new application with C++ activity
2. Modify the app to create and delete a JNI reference, so that there is a local variable in the method scope that contains an already-deleted JNI reference, e.g.,:

```
extern "C" JNIEXPORT jstring JNICALL
Java_com_example_myapplication_MainActivity_stringFromJNI(
        JNIEnv* env,
        jobject /* this */) {
    std::string hello = "Hello from C++";
    jstring localStr = env->NewStringUTF(hello.c_str());
    jstring globalStr = reinterpret_cast<jstring>(env->NewGlobalRef(localStr));
    if (globalStr == nullptr) {
        std::cerr << "ERROR: cannot convert to global!";
        exit(1);
    }
    env->DeleteGlobalRef(globalStr); // SET 1ST BREAKPOINT HERE.
    return localStr;  // SET 2ND BREAKPOINT HERE.
}
```

3. Set breakpoints at the statements above.
4. Use Auto/Dual debugger to start debugging.
5. At the 1st breakpoint, open Logcat and clear the window.
6. Go back to debugger and hit Continue.
7. When the 2nd breakpoint hits, observe the logcat window contents.

Expected result:

- no errors in logcat.

Actual result:

```
2022-02-28 09:41:36.452 4783-4783/com.example.myapplication E/e.myapplicatio: JNI ERROR (app bug): accessed stale Global 0x286a  (index 646 in a table of siz
2022-02-28 09:41:36.452 4783-4783/com.example.myapplication A/e.myapplicatio: java_vm_ext.cc:570] JNI DETECTED ERROR IN APPLICATION: use of deleted global re
2022-02-28 09:41:36.452 4783-4783/com.example.myapplication A/e.myapplicatio: java_vm_ext.cc:570]     from java.lang.String com.example.myapplication.MainAct
```

Android Studio C++ debugger uses the `jstring_reader.py` LLDB script to extract the internal string in all `jstring`-type variables. However, this script does not check if the target `jstring` has when it calls the ART method to read the internals of a deleted `jstring`, the operation fails, and prints those errors to logcat.

✓ Links (1)

Links (1)

"The good news is that, these messages are not printed when using API Level 30+ due to this change which converted them from being logs to DCHECKs."

COMMENTS All comments

em...@google.com <em...@google.com> #2

The good news is that, these messages are not printed when using API Level 30+ due to this change which converted them from being logs to `DCHECK`s.

On the other hand, I'd like to verify what happens when using the debug version of libart (i.e. `libartd.so`). If it causes the app to crash, then we should either convert those `DCHECK`s to som new API in ART, such as `IsReferenceValid(jobject)` or `GetReferenceType(jobject)` (that also allows passing deleted references), so that we can verify a reference before calling `Deco`

it.

em...@google.com <em...@google.com> #3 M

BTW, in Android Studio, you can always disable the problematic `jstring` formatter to avoid this problem for all Android API levels:

```
type category disable "JNI types"
```

Message last modified on Mar 9, 2022 03:32AM

em...@google.com <em...@google.com>

Assigned to em...@google.com.

em...@google.com <em...@google.com> #4 M

Status: Won't Fix (Infeasible)

A conversation with the art-team did not lead to a result. They suggested we always use LLDB to intercept the SIGABRT signals if an expression evaluation fails, but I believe that is

- neither necessary: because IMO there is no way the release version of DecodeJObject will start sending SIGABRT in the future
- nor helpful: it will not prevent spurious logs being printed to logcat

Since there won't be any logs printed for API 30+, I'll close this bug as will-not-fix.