Android Public Tracker > App Development > Android Studio > Deployment > C++ Debugger 198464696

Cannot debug native methods with jstring parameters

+1 Hotlists Mark as Duplicate

Comments (8) Dependencies Duplicates (0) Blocking (0) Resources (2)

Fixed Bug P2 + Add Hotlist

STATUS UPDATE No update yet. Edit

DESCRIPTION ly...@gmail.com created issue #1 Sep 2, 2021 09:35PM

Code:

```
extern "C"
JNIEXPORT void JNICALL
Java_com_example_myapplication_MainActivity_nGetHeight(JNIEnv *env, jobject thiz, jstring text) {

    std::string te = "aaaa";  // Breakpoint at this line
    printf("%s", te.c_str());
}
```

Env:

Android Studio Arctic Fox | 2020.3.1 Patch 1 Build #AI-203.7717.56.2031.7621141, built on August 8, 2021 Runtime version: 11.0.10+0-b96-7249189 amd64 VM: OpenJDK 64-Bit Server VM by Oracle Corporation Windows 10 10.0 GC: G1 Young Generation, G1 Old Generation Memory: 2048M Cores: 8 Registry: external.system.auto.import.disabled=true, debugger.watches.in.variables=false Non-Bundled Plugins: org.jetbrains.kotlin, org.intellij.plugins.markdown

ndkVersion "23.0.7599858"

Android Emulator: 30.8.4

System Image: Google Apis Intel x86

✓ Mentioned issues (1) ✓ Links (1) Hide all

Mentioned issues (1)

P2 App crash (segfault) when debugging JNI functions with local jstring reference "Related bug: b/120865679" em...@ #5

Links (1)

" ...e change http://ag/11339592 (internal-only link) should have fixed this problem, but I think we are packaging the wrong jstring_reader.py into Android ... " em...@ #4

COMMENTS

All comments ▾ ↓ Oldest first

en...@google.com <en...@google.com> #2 Sep 3, 2021 06:27AM

Status: New

works for me in 4.1.2 ... are you saying this is an issue specific to the Studio preview release?

over to the Studio folks anyway, since this doesn't seem NDK related...

vs...@google.com <vs...@google.com> Sep 8, 2021 01:40AM

Assigned to an...@google.com.

gi...@google.com <gi...@google.com> Sep 9, 2021 08:11AM

Reassigned to em...@google.com.

em...@google.com <em...@google.com> #3 Sep 11, 2021 03:16AM

Accepted by em...@google.com.

On both Arctic Fox Patch 2 and Bumblebee Canary 10, I observe the debugger crashes when it encounters a `jstring`.

While we investigate the root cause, you can work around this crash by adding the following to your "LLDB Post Attach Commands": `type category disable "JNI types"`. See screenshot.

Message last modified on Sep 11, 2021 03:17AM

Fri Sep 10 2021 10:16:12 GMT-0700 (Pacific Daylight Time).png

Reporter	
Type	Bu
Priority	P2
Severity	S2
Status	Fi
Access	De
Assignee	
Verifier	--
Collaborators	
CC	en ly..
AOSP ID	--
Blocking Release	--
Release Status	--
Found In	--
Targeted To	--
Verified In	--
In Prod	

Show 1 addition

32 KB View Download

em...@google.com <em...@google.com> #4 Sep 11, 2021 03:29AM

~~The change http://ag/11339592 (internal-only link) should have fixed this problem, but I think we are packaging the wrong `jstring_reader.py` into Android Studio.~~ This was probably not relevant.

Message last modified on Sep 11, 2021 03:33AM

em...@google.com <em...@google.com> #5 Sep 11, 2021 03:30AM

Related bug: b/120865679

em...@google.com <em...@google.com> #6 Sep 11, 2021 08:48AM

A side note for future investigations:

- In our `jstring` pretty printers, LLDB invokes this expression: `art::Thread::DecodeJObject(art::Thread::CurrentFromGdb(), jstring_object)`
- In `env->GetStringUTFChars(jstring_object, 0)`, the inner expression becomes: `art::Thread::DecodeJObject(ThreadForEnv(env), jstring_object)` (which seems to be the same thread?)

Message last modified on Sep 11, 2021 09:02AM

em...@google.com <em...@google.com> #7 Sep 11, 2021 09:43AM

The bottom line here is, yes there's a problem with `jstring` evaluation on Android-x86 due to a mismatch between `libart` using `SIGSEGV` for special internal operations and LLDB's expression evaluator panicking when it sees such a SEGV signal.

We will add a workaround into Android Studio that disables `jstring` formatters for `x86` to avoid unexpected triggering of this bug.

Further work to fix the root cause of this mismatch is not planned. If possible, you can use `x86_64` which doesn't suffer from this problem.

em...@google.com <em...@google.com> #8 Sep 15, 2021 01:05AM

Marked as fixed.

Workaround is added. It will be included in Android Studio C release (whatever comes after bumblebee).