Vendor extensions sometimes cause crash

STATUS UPDATE No update yet. Edit

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

CAMERAX VERSION : 1.3.0-alpha06

ANDROID OS: Android 12

DEVICE NAME:

Device name	Affected users
samsung beyond1	488
samsung beyond0	237
samsung beyond2	221
samsung a53x	63
samsung d2s	34
samsung r8q	26
samsung o1s	21
samsung d1	19
samsung r9q	18
samsung r8s	15
samsung x1s	12
samsung rOs	11
samsung a52sxq	10
samsung a52q	8
samsung d2x	7
samsung y2s	6
samsung bOs	5
samsung p3s	5
samsung gOs	5
samsung t2s	5
samsung beyond1q	4
samsung beyond0q	3
samsung d2q	3
samsung a42xq	3
samsung c2s	3
samsung beyond2q	2
samsung z3s	2
samsung b4q	2
samsung r3q	1
samsung winner	1
OUKITEL C32	1
realme RE87BAL1	1
Nokia SFI	1
OSCAL C80	1
realme RE5894	1
samsung a52xq	1
samsung a72q	1
samsung b2q	1
samsung c1s	1
samsung r5q	1
samsung r7	1

Device name	Affected users
Total	**1251**

DESCRIPTION:

After deploying our App with enabled Vendor Extensions in cameraX we are observing a crash with stack trace in the Play Console.

LIST ANY EXPERIMENTAL FEATURES: @ExperimentalCamera2Interop

STEPS TO REPRODUCE:

1. Take a picture with enabled Vendor Extension AUTO
2. Sometimes the app is crashing with a stack trace

OBSERVED RESULTS: Stack trace in Play Console

```
JNI DETECTED ERROR IN APPLICATION: buffer capacity greater than maximum jint: 3657424885

backtrace:

  #00  pc 0x00000000000513c4  /apex/com.android.runtime/lib64/bionic/libc.so (abort+180)

  #01  pc 0x000000000062f7b0  /apex/com.android.art/lib64/libart.so (art::runtime::Abort(char const*)+728)

  #02  pc 0x0000000000015aa0  /system/lib64/libbase.so (android::base::SetAborter(std::1::function<void (char const*)>&&)::$_3::invoke(char const*)+80)

  #03  pc 0x000000000001508c  /system/lib64/libbase.so (android::base::logmessage::~LogMessage()+364)

  #04  pc 0x0000000000454b84  /apex/com.android.art/lib64/libart.so (art::javavmext::JniAbort(char const, char const)+2580)

  #05  pc 0x0000000000454cf0  /apex/com.android.art/lib64/libart.so (art::javavmext::JniAbortF(char const, char const, ...)+184)

  #06  pc 0x00000000004c4260  /apex/com.android.art/lib64/libart.so (art::JNI<false>::NewDirectByteBuffer(_JNIEnv, void, long)+120)

  #07  pc 0x0000000000043714  /system/lib64/libmedia_jni.so (Image_createSurfacePlanes(_JNIEnv, _jobject, int, int, unsigned long)+940)

  #08  pc 0x000000000024b960  /system/framework/arm64/boot-framework.oat (art_jni_trampoline+128)

  #09  pc 0x0000000000212b80  /apex/com.android.art/lib64/libart.so (nterp_helper+5648)

  #10  pc 0x000000000034c09a  /system/framework/framework.jar (android.media.ImageReader$SurfaceImage.getPlanes+50)

  #11  pc 0x0000000000218964  /apex/com.android.art/lib64/libart.so (art_quick_invoke_stub+548)

  #12  pc 0x00000000002851f0  /apex/com.android.art/lib64/libart.so (art::artmethod::Invoke(art::Thread, unsigned int, unsigned int, art::JValue, char const)

  #13  pc 0x00000000003e70f0  /apex/com.android.art/lib64/libart.so (art::interpreter::ArtInterpreterToCompiledCodeBridge(art::Thread, art::ArtMethod, art::S

  #14  pc 0x00000000003e26c8  /apex/com.android.art/lib64/libart.so (bool art::interpreter::DoCall<false, true>(art::ArtMethod, art::Thread, art::ShadowFrame

  #15  pc 0x0000000000758138  /apex/com.android.art/lib64/libart.so (MterpInvokeVirtual+2144)

  #16  pc 0x0000000000203814  /apex/com.android.art/lib64/libart.so (mterp_op_invoke_virtual+20)

  #17  pc 0x00000000002965d8  /data/app/~~kbiLXYoOtmb2jSMeGg5Vkw==/com.floraincognita.app.floraincognita-8FK1qloWxWUKaIthRznycg==/oat/arm64/base.vdex (androi

  #18  pc 0x00000000003d98e0  /apex/com.android.art/lib64/libart.so (art::interpreter::Execute(art::Thread*, art::CodeItemDataAccessor const&, art::ShadowFra

  #19  pc 0x00000000003e1254  /apex/com.android.art/lib64/libart.so (art::interpreter::ArtInterpreterToInterpreterBridge(art::Thread, art::CodeItemDataAccess

  #20  pc 0x00000000003e26ac  /apex/com.android.art/lib64/libart.so (bool art::interpreter::DoCall<false, true>(art::ArtMethod, art::Thread, art::ShadowFrame

  #21  pc 0x00000000007610b4  /apex/com.android.art/lib64/libart.so (MterpInvokeDirect+1068)

  #22  pc 0x0000000000203914  /apex/com.android.art/lib64/libart.so (mterp_op_invoke_direct+20)

  #23  pc 0x00000000002966ca  /data/app/~~kbiLXYoOtmb2jSMeGg5Vkw==/com.floraincognita.app.floraincognita-8FK1qloWxWUKaIthRznycg==/oat/arm64/base.vdex (androi

  #24  pc 0x00000000003d98e0  /apex/com.android.art/lib64/libart.so (art::interpreter::Execute(art::Thread*, art::CodeItemDataAccessor const&, art::ShadowFra

  #25  pc 0x00000000003e1254  /apex/com.android.art/lib64/libart.so (art::interpreter::ArtInterpreterToInterpreterBridge(art::Thread, art::CodeItemDataAccess

  #26  pc 0x00000000003e26ac  /apex/com.android.art/lib64/libart.so (bool art::interpreter::DoCall<false, true>(art::ArtMethod, art::Thread, art::ShadowFrame

  #27  pc 0x000000000075e3a8  /apex/com.android.art/lib64/libart.so (MterpInvokeInterface+2536)

  #28  pc 0x0000000000203a14  /apex/com.android.art/lib64/libart.so (mterp_op_invoke_interface+20)

  #29  pc 0x000000000029f65e  /data/app/~~kbiLXYoOtmb2jSMeGg5Vkw==/com.floraincognita.app.floraincognita-8FK1qloWxWUKaIthRznycg==/oat/arm64/base.vdex (androi

  #30  pc 0x00000000003d98e0  /apex/com.android.art/lib64/libart.so (art::interpreter::Execute(art::Thread*, art::CodeItemDataAccessor const&, art::ShadowFra
  #31  pc 0x000000000074657c  /apex/com.android.art/lib64/libart.so (artQuickToInterpreterBridge+780)

  #32  pc 0x0000000000222378  /apex/com.android.art/lib64/libart.so (art_quick_to_interpreter_bridge+88)

  #33  pc 0x0000000000213390  /apex/com.android.art/lib64/libart.so (nterp_helper+7712)

  #34  pc 0x000000000029c3a6  /data/app/~~kbiLXYoOtmb2jSMeGg5Vkw==/com.floraincognita.app.floraincognita-8FK1qloWxWUKaIthRznycg==/oat/arm64/base.vdex (androi

  #35  pc 0x0000000000218be8  /apex/com.android.art/lib64/libart.so (art_quick_invoke_static_stub+568)

  #36  pc 0x000000000028520c  /apex/com.android.art/lib64/libart.so (art::artmethod::Invoke(art::Thread, unsigned int, unsigned int, art::JValue, char const)

  #37  pc 0x00000000003e70f0  /apex/com.android.art/lib64/libart.so (art::interpreter::ArtInterpreterToCompiledCodeBridge(art::Thread, art::ArtMethod, art::S

  #38  pc 0x00000000003e26c8  /apex/com.android.art/lib64/libart.so (bool art::interpreter::DoCall<false, true>(art::ArtMethod, art::Thread, art::ShadowFrame

  #39  pc 0x0000000000762df4  /apex/com.android.art/lib64/libart.so (MterpInvokeStatic+1012)

  #40  pc 0x0000000000203994  /apex/com.android.art/lib64/libart.so (mterp_op_invoke_static+20)

  #41  pc 0x000000000029611c  /data/app/~~kbiLXYoOtmb2jSMeGg5Vkw==/com.floraincognita.app.floraincognita-8FK1qloWxWUKaIthRznycg==/oat/arm64/base.vdex (androi

  #42  pc 0x00000000003d98e0  /apex/com.android.art/lib64/libart.so (art::interpreter::Execute(art::Thread*, art::CodeItemDataAccessor const&, art::ShadowFra

  #43  pc 0x00000000003e1254  /apex/com.android.art/lib64/libart.so (art::interpreter::ArtInterpreterToInterpreterBridge(art::Thread, art::CodeItemDataAccess

  #44  pc 0x00000000003e26ac  /apex/com.android.art/lib64/libart.so (bool art::interpreter::DoCall<false, true>(art::ArtMethod, art::Thread, art::ShadowFrame

  #45  pc 0x000000000075e3a8  /apex/com.android.art/lib64/libart.so (MterpInvokeInterface+2536)

  #46  pc 0x0000000000203a14  /apex/com.android.art/lib64/libart.so (mterp_op_invoke_interface+20)

  #47  pc 0x000000000029f910  /data/app/~~kbiLXYoOtmb2jSMeGg5Vkw==/com.floraincognita.app.floraincognita-8FK1qloWxWUKaIthRznycg==/oat/arm64/base.vdex (androi

  #48  pc 0x00000000003d98e0  /apex/com.android.art/lib64/libart.so (art::interpreter::Execute(art::Thread*, art::CodeItemDataAccessor const&, art::ShadowFra

  #49  pc 0x00000000003e1254  /apex/com.android.art/lib64/libart.so (art::interpreter::ArtInterpreterToInterpreterBridge(art::Thread, art::CodeItemDataAccess

  #50  pc 0x00000000003e26ac  /apex/com.android.art/lib64/libart.so (bool art::interpreter::DoCall<false, true>(art::ArtMethod, art::Thread, art::ShadowFrame

  #51  pc 0x0000000000758138  /apex/com.android.art/lib64/libart.so (MterpInvokeVirtual+2144)

  #52  pc 0x0000000000203814  /apex/com.android.art/lib64/libart.so (mterp_op_invoke_virtual+20)

  #53  pc 0x0000000000296320  /data/app/~~kbiLXYoOtmb2jSMeGg5Vkw==/com.floraincognita.app.floraincognita-8FK1qloWxWUKaIthRznycg==/oat/arm64/base.vdex (androi

  #54  pc 0x00000000003d98e0  /apex/com.android.art/lib64/libart.so (art::interpreter::Execute(art::Thread*, art::CodeItemDataAccessor const&, art::ShadowFra

  #55  pc 0x00000000003e1254  /apex/com.android.art/lib64/libart.so (art::interpreter::ArtInterpreterToInterpreterBridge(art::Thread, art::CodeItemDataAccess

  #56  pc 0x00000000003e26ac  /apex/com.android.art/lib64/libart.so (bool art::interpreter::DoCall<false, true>(art::ArtMethod, art::Thread, art::ShadowFrame

  #57  pc 0x000000000075e3a8  /apex/com.android.art/lib64/libart.so (MterpInvokeInterface+2536)

  #58  pc 0x0000000000203a14  /apex/com.android.art/lib64/libart.so (mterp_op_invoke_interface+20)

  #59  pc 0x00000000002968ac  /data/app/~~kbiLXYoOtmb2jSMeGg5Vkw==/com.floraincognita.app.floraincognita-8FK1qloWxWUKaIthRznycg==/oat/arm64/base.vdex (androi

  #60  pc 0x00000000003d98e0  /apex/com.android.art/lib64/libart.so (art::interpreter::Execute(art::Thread*, art::CodeItemDataAccessor const&, art::ShadowFra

  #61  pc 0x00000000003e1254  /apex/com.android.art/lib64/libart.so (art::interpreter::ArtInterpreterToInterpreterBridge(art::Thread, art::CodeItemDataAccess

  #62  pc 0x00000000003e26ac  /apex/com.android.art/lib64/libart.so (bool art::interpreter::DoCall<false, true>(art::ArtMethod, art::Thread, art::ShadowFrame

  #63  pc 0x0000000000758138  /apex/com.android.art/lib64/libart.so (MterpInvokeVirtual+2144)

  #64  pc 0x0000000000203814  /apex/com.android.art/lib64/libart.so (mterp_op_invoke_virtual+20)

  #65  pc 0x0000000000295c66  /data/app/~~kbiLXYoOtmb2jSMeGg5Vkw==/com.floraincognita.app.floraincognita-8FK1qloWxWUKaIthRznycg==/oat/arm64/base.vdex (androi

  #66  pc 0x00000000003d98e0  /apex/com.android.art/lib64/libart.so (art::interpreter::Execute(art::Thread*, art::CodeItemDataAccessor const&, art::ShadowFra

  #67  pc 0x000000000074657c  /apex/com.android.art/lib64/libart.so (artQuickToInterpreterBridge+780)

  #68  pc 0x0000000000222378  /apex/com.android.art/lib64/libart.so (art_quick_to_interpreter_bridge+88)

  #69  pc 0x000000000078d870  /system/framework/arm64/boot-framework.oat (android.os.Handler.dispatchMessage+80)

  #70  pc 0x0000000000790b2c  /system/framework/arm64/boot-framework.oat (android.os.Looper.loopOnce+1036)
```

```
#71  pc 0x0000000000790684  /system/framework/arm64/boot-framework.oat (android.os.Looper.loop+516)

#72  pc 0x000000000050d940  /system/framework/arm64/boot-framework.oat (android.app.ActivityThread.main+800)

#73  pc 0x0000000000218be8  /apex/com.android.art/lib64/libart.so (art_quick_invoke_static_stub+568)

#74  pc 0x000000000028520c  /apex/com.android.art/lib64/libart.so (art::artmethod::Invoke(art::Thread, unsigned int, unsigned int, art::JValue, char const)

#75  pc 0x0000000000627760  /apex/com.android.art/lib64/libart.so (_jobject* art::InvokeMethod<(art::PointerSize)8>(art::ScopedObjectAccessAlreadyRunnable

#76  pc 0x0000000000597a38  /apex/com.android.art/lib64/libart.so (art::Method_invoke(_JNIEnv, _jobject, _jobject, _jobjectArray)+48)

#77  pc 0x00000000000b2f74  /apex/com.android.art/javalib/arm64/boot.oat (art_jni_trampoline+132)

#78  pc 0x0000000000ae300c  /system/framework/arm64/boot-framework.oat (com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run+140)

#79  pc 0x0000000000aec438  /system/framework/arm64/boot-framework.oat (com.android.internal.os.ZygoteInit.main+2376)

#80  pc 0x0000000000218be8  /apex/com.android.art/lib64/libart.so (art_quick_invoke_static_stub+568)

#81  pc 0x000000000028520c  /apex/com.android.art/lib64/libart.so (art::artmethod::Invoke(art::Thread, unsigned int, unsigned int, art::JValue, char const)

#82  pc 0x0000000000627ec0  /apex/com.android.art/lib64/libart.so (art::JValue art::InvokeWithVarArgs<art::ArtMethod>(art::ScopedObjectAccessAlreadyRunnabl

#83  pc 0x0000000000628394  /apex/com.android.art/lib64/libart.so (art::JValue art::InvokeWithVarArgs<_jmethodID>(art::ScopedObjectAccessAlreadyRunnable co

#84  pc 0x0000000000501094  /apex/com.android.art/lib64/libart.so (art::JNI<true>::CallStaticVoidMethodV(_JNIEnv, _jclass, _jmethodID*, std::__va_list)+612

#85  pc 0x00000000000b2b28  /system/lib64/libandroid_runtime.so (_JNIEnv::CallStaticVoidMethod(_jclass, _jmethodID, ...)+120)

#86  pc 0x00000000000becd8  /system/lib64/libandroid_runtime.so (android::androidruntime::start(char const*, android::Vector<android::String8> const&, bool

#87  pc 0x00000000000025b0  /system/bin/app_process64 (main+1368)

#88  pc 0x0000000000049b48  /apex/com.android.runtime/lib64/bionic/libc.so (__libc_init+96)
```

EXPECTED RESULTS: Runs without crash

REPRODUCIBILITY: 1 of 20

ADDITIONAL INFORMATION:

When using Vendor Extensions the ImageFormat is locked to JPEG. Normally our app uses YUV_420_888 as ImageFormat .

To process the JPEG we are calling this line in Java:

```
ImageProxy image = (ImageProxy from ImageCapture.OnImageCapturedCallback());
ByteBuffer buffer = image.getPlanes()[0].getBuffer();
```

This line leads to the Stack trace observed from the Play Console.

✓ **Links (1)**

Links (1)

"The issue was fixed by CL https://android-review.googlesource.com/c/platform/frameworks/support/+/2674836 The next CameraX release will contain the fix."

COMMENTS

er...@google.com <er...@google.com>

Reassigned to ja...@google.com.

jo...@gmail.com <jo...@gmail.com> #2

Just for the sake of completeness:

The observed number of occurrences is measured per month and is derived from the reports in the Play Console. Therefore, only users who have opted in to report this issue are taken into a

sc...@google.com <sc...@google.com> #3

Reassigned to sc...@google.com.

We are able to reproduce the issue on Android 12 devices. Will fix it as soon as possible. Thanks for reporting the issues.

sc...@google.com <sc...@google.com>

Status: Duplicate of 289021003

jo...@gmail.com <jo...@gmail.com> #4

Unfortunately, I am unable to see the mentioned issue. Would it be possible to grant me the necessary rights to view this issue, so that I can track its progress?

sc...@google.com <sc...@google.com> #5

No worry, this is just for internal tracking. The issue has been fixed and we are verifying it through our lab testing. will keep you posted here.

sc...@google.com <sc...@google.com> #6

The issue was fixed by CL https://android-review.googlesource.com/c/platform/frameworks/support/+/2674836 The next CameraX release will contain the fix.