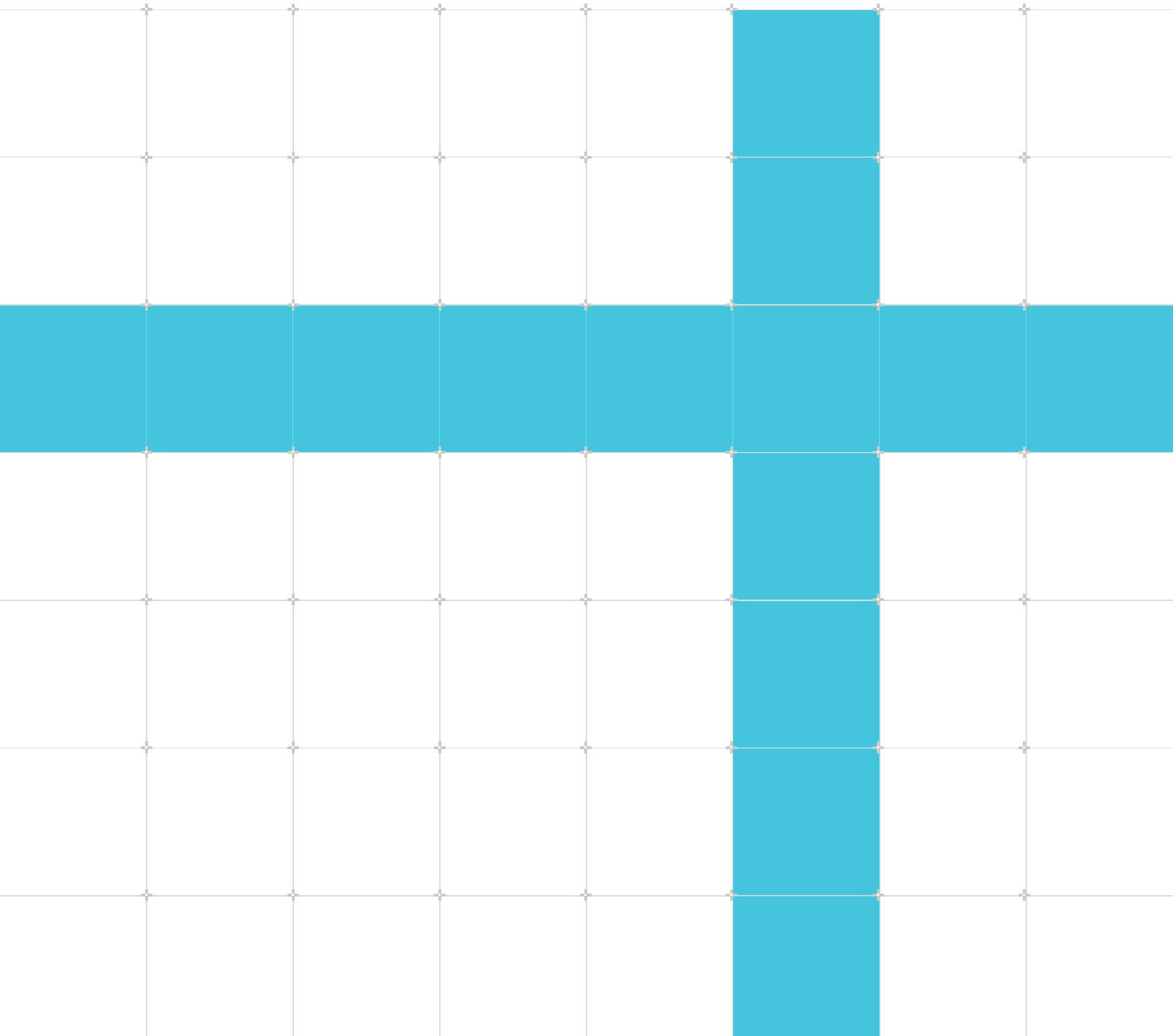
Arm Mobile Studio

1.2

FAQs

Issue 00
102718_0102_00_en



Arm Mobile Studio
FAQs

Release information

Document history

Issue	Date	Confidentiality	Change
0102-00	28 June 2022	Non-Confidential	LWI directory update

Proprietary Notice

and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Feedback

Arm® welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Contents

1. Arm Mobile Studio FAQs

Here are answers to common questions about Arm Mobile Studio.

Please ensure you are using the latest version of Arm Mobile Studio, which you can download here.

2. Arm Mobile Studio tools fail with a license error

If licensing is not set correctly, Arm Mobile Studio tools will fail with a licensing error. The possible causes for this are described here.

Arm Mobile Studio license has expired

Arm Mobile Studio 2019.x Starter Edition licenses have expired. Please download and install the latest version of Arm Mobile Studio to resolve this.

Environment variables are not set to use an upgraded license file

Arm Mobile Studio ships with an integrated Starter Edition license that is automatically detected. An alternative license, such as a commercial license, can be specified using environment variables. Environment variables take precedence over the integrated license configuration, so any misconfiguration can result in a license error.

Solution

1. Check that the `ARMLMD_LICENSE_FILE` environment variable is set correctly. This variable should specify the path to a valid Mobile Studio license file (node-locked license) or license server (floating licenses).
2. Check that the `ARM_MS_TOOL_VARIANT` environment variable is set to `ms_evaluation`.
3. Check that the optional `ARM_MS_PRODUCT_DEF` environment variable specifies a license map that matches a Mobile Studio edition that is available in the license file. If you are unsure, unset this variable.

To revert to the Starter Edition license provided in your Arm Mobile Studio installation, unset all of the licensing environment variables:

- `ARMLMD_LICENSE_FILE`
- `ARM_MS_TOOL_VARIANT`
- `ARM_MS_PRODUCT_DEF`

Arm Development Studio licenses

Streamline and Graphics Analyzer ship in both Arm Mobile Studio and Arm Development Studio, but the builds of the tool for the two studios are different. You cannot use a Mobile Studio or Development Studio build of a tool with a license for the other build. For more information about Development Studio license configuration, see Licensing Arm Development Studio.

Related information

- More FAQs
- Ask a question on the Arm Community forum

3. Cannot import Performance Advisor JSON into ELK

When trying to import JSON reports that Performance Advisor has generated into an Elasticsearch database, the import fails.

Condition

Affected releases: 2020.1

Cause

Elasticsearch does not process JSON files that are pretty-printed. It also expects an index statement at the start of the JSON file, and a newline at the end. In Arm Mobile Studio 2020.1 and earlier, JSON files are pretty-printed by default, and contain only the JSON data fields for the reported performance data.

Workaround

Download this Python script and run it on your JSON files to convert them to an acceptable format for Elasticsearch:

```
python3 prepare-json-for-elastic-search.py filename.json
```

Alternatively, make the following changes to your JSON files:

1. Elasticsearch expects the following on line 1 of a JSON file:

   ```
   {"index":{}}
   ```

   Alternatively, you can add your own values to tell Elasticsearch which index to add the file to, and an id number:

   ```
   { "index" : { "_index" : "test", "_id" : "1" } }
   ```

   If you don't specify an index you will need to specify it with the command you use to import the file, for example:

   ```
   curl -X POST "<Elasticsearch_location>/indexname/_bulk?pretty" -H 'Content-Type:
    application/x-ndjson' --data-binary @<file_name>.json
   ```

   If you don't specify an ID number, Elasticsearch allocates a random ID to the document.
2. On line 2, Elasticsearch expects the JSON data with no newlines.
3. Add a newline at the end of the doc (line 3). The file should look like the following:

   ```
   {"index":{}}
   {JSON file contents}
   ```

```
\n
```

Related information

- Elasticsearch Bulk API documentation
- Generate a JSON file
- More FAQs
- Ask a question on the Arm Community forum.

4. Device connection issues

In some cases, Arm Mobile Studio tools can not find the connected Android device. Here are some common reasons why this happens, and how to resolve them.

Condition

Affected releases: All.

Check your device is set up correctly

Do the following:

1. Update the firmware on your device to the latest version.
2. Check that the device is switched on, connected to your host machine through USB, and is running Android 8 or later.
3. Ensure Developer Mode is enabled on the device, and enable USB Debugging using Settings > Developer options.
4. Type `adb devices` into a terminal on your host machine, to return the ID of any connected devices.

Check that ADB is installed

Check that you have Android Debug Bridge (ADB) installed and working correctly:

1. In a command terminal, type `adb --help` to check that ADB is installed - this should return the list of options for the `adb` command.

   If you can not run the `adb` command successfully, it may not be installed, or you may not be able to run it from your current location.
2. Install Android Debug Bridge (ADB). ADB is available with the Android SDK platform tools, which are installed as part of Android Studio, or you can download them separately here.
3. Edit your `PATH` environment variable to add the path to the ADB executable, so that you can run the `adb` command from any directory.

Errors running the connection script

Streamline and Performance Advisor use a connection script to set up the device ready for capture. Use either the `gator_me.py` script, or the `lwi_me.py` script as described in the Get started guides to do this. If the script fails, check the following:

1. Ensure that Python 3.5 or later is installed, and the path to the Python executable is added to your `PATH` environment variable, so that you can run Python commands from any directory. To test Python, in a terminal, type `python --help` which returns the list of available command options.
2. When running the script, ensure that either:

a. You have used the `--daemon` option to specify the path to the daemon application `gatord`, that will be used to collect data from your device.

```
python3 {gator_me.py|lwi_me.py} --daemon <path_to_gatord>
```

b. You have copied the required `gatord` file into the current directory.

This file is provided in your Arm Mobile Studio installation directory: `<install_directory>/streamline/bin/{arm|arm64}/gatord`. Different files are supplied for 32-bit (`arm`) or 64-bit (`arm64`) applications.

3. When running Performance Advisor's `lwi_me.py` script, you need to specify the `--lwi-out-dir` option to provide the path to an empty directory where Performance Advisor will store the frame images it captures.

```
python3 lwi_me.py --daemon <path_to_gatord> --lwi-out-dir
 <path_to_empty_directory>
```

4. To use Performance Advisor's `lwi_me.py` script to connect to Android 10 devices, you need to specify the `-lwi-gles-layer-lib-path` or `--lwi-vk-layer-lib-path` option, to provide the path to the OpenGL ES or Vulkan layer library file.

```
python3 lwi_me.py --daemon <path_to_gatord> --lwi-out-dir <path_to_empty_directory>
[--lwi-gles-layer-lib-path | --lwi-vk-layer-lib-path <path_to_Android10_layer_lib>]
```

This file is provided in your Arm Mobile Studio installation directory: `<install_directory>/performance_advisor/bin/android/{arm|arm64}`. Different files are supplied for rooted or unrooted devices, and for 32-bit (`armeabi-v7a`) or 64-bit (`arm64-v8a`) applications.

5. Ensure you are using the latest version of Mobile Studio.

Device support

As Arm-based processors are very widespread in the smartphone industry it is impossible for us to test our tools on all devices available for sale to the public. Check our supported devices page for a list of devices we have tested and confirm will work with Arm Mobile Studio.

Get help

If you're still experiencing connection issues, ask for help on the Arm Community graphics forum.

Related information

- More FAQs
- Ask a question on the Arm Community forum.

5. Streamline can not access my device

When launching Streamline, the connected device is not showing in the Start tab.

Condition

Affects versions: 2020.2 and earlier

Cause

In Arm Mobile Studio versions prior to 2020.3, it was necessary to use a Python script to connect Streamline to your device.

The Python script, `gator_me.py` installs a daemon, `gatord`, on your device. Streamline uses this daemon to connect to unrooted Android devices and collect data. Follow these steps to run the script so that Streamline can communicate with your device.

Workaround

Do the following:

1. In a terminal window, check that the device is accessible by typing:

   ```
   adb devices
   ```

   This should return the ID of your device. If it does not, check that you have installed Android Debug Bridge and connected the device via USB.
2. On your host machine, navigate to the Streamline installation directory, `<install_dir>/streamline/gator/` where you will find the `gator_me.py` script.
3. Run the `gator_me.py` script with the `--daemon` option, to supply the path to the `gatord` binary that will be installed on the device. For example:

   ```
   python3 gator_me.py --daemon ../bin/arm64/gatord
   ```

   There are two versions of the `gatord`, for 32-bit or 64-bit architectures, located in the following directories:

   - `<install_dir>/streamline/bin/arm64/` for Armv8 64-bit architectures.
   - `<install_dir>/streamline/bin/arm/` for 32-bit architectures.
4. The script will return a numbered list of the Android package names for the debuggable applications that are installed on your device. Enter the number of the package you want to profile.

   Alternatively, if you know the Android package name of the app you want to profile you can specify it when running the script, using the `--package` option.

   ```
   python3 gator_me.py --package com.mycompany.myapp --daemon ../bin/arm64/gatord
   ```
5. Launch Streamline:

a. On Windows, from the Start menu, navigate to the Arm Mobile Studio folder, and select the Streamline shortcut.

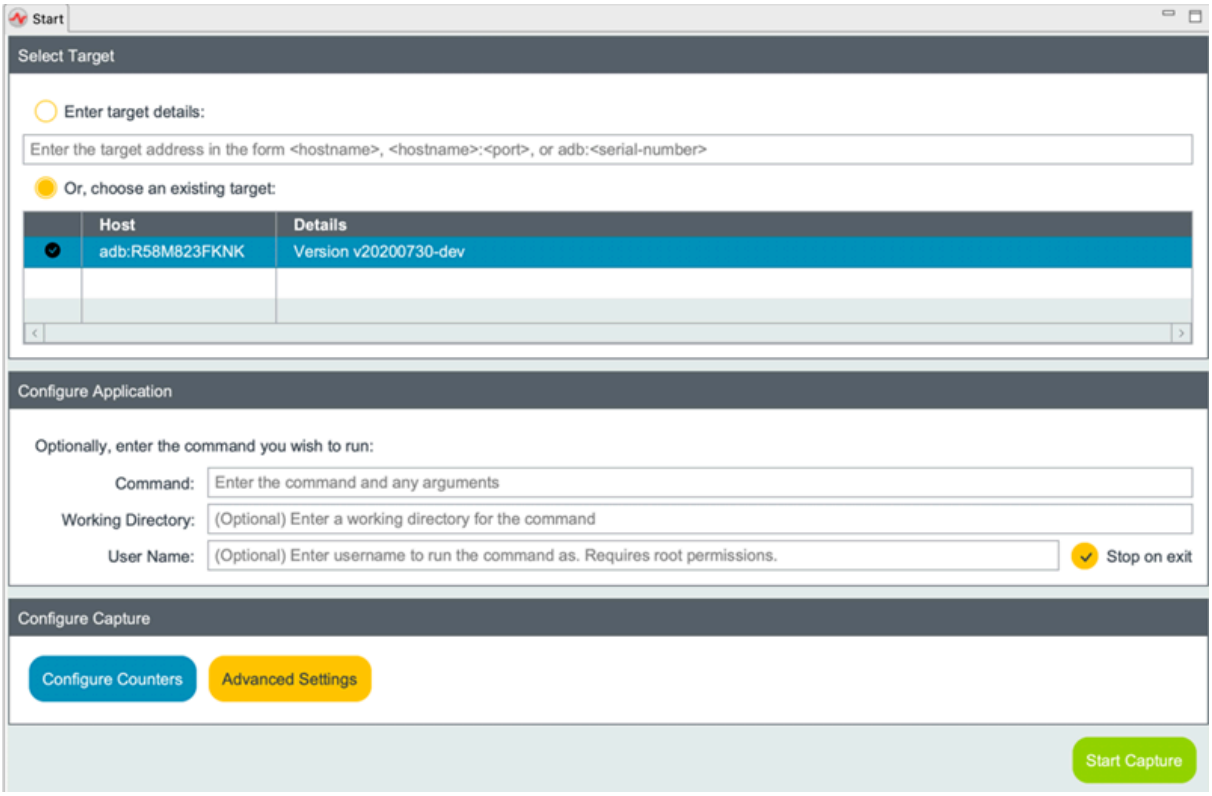
b. On macOS, go to the `<install_dir>/streamline` folder, and double-click the `Streamline.app` file.

c. On Linux, go to the `<install_dir>/streamline` folder, and run the `Streamline` file:

```
cd <install_dir>/streamline
   ./Streamline
```

6. Use the Start tab in Streamline to select your device. Select Or, choose an existing target, then select your device from the list.

Figure 5-1: Connect to your device in Streamline



7. Before starting the capture, choose Configure Counters to choose a counter template.

Note

IMPORTANT: When you've finished capturing data from this device, switch back to the terminal running the `gator_me.py` script and press any key to terminate it. The script kills all processes that it started and removes gatord from the target.

Related information

- More FAQs

- Ask a question on the Arm Community forum.

6. Error downloading Arm Mobile Studio package

In some browser versions, when logged into your Arm account and you attempt to download the Arm Mobile Studio package, the download fails with the following error:

```
Bad Request Your browser sent a request that this server could not understand. Size
 of a request header field exceeds server limit.
```

Cause

This is a known website issue that we are working on to resolve.

Workaround

To workaround this issue:

- Clear cookies and try the download again.
- Use an incognito window in your browser and try the download again.

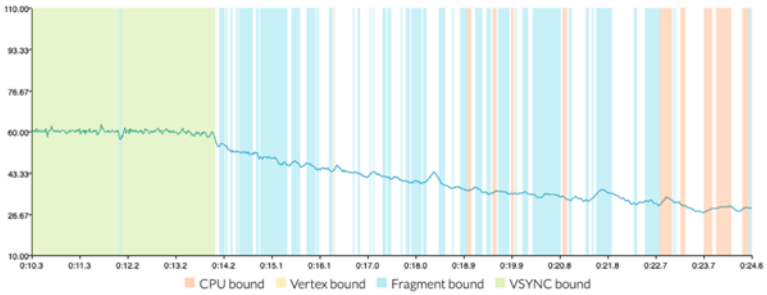
Related information

- More FAQs
- Ask a question on the Arm Community forum.

7. Blank sections in Performance Advisor FPS chart

When generating a report with Performance Advisor, some sections of the FPS chart are shown in white, instead of showing a color to indicate the type of boundness: CPU-bound, fragment-bound, vertex-bound, or performing well (VSYNC).

Figure 7-1: FPS summary chart showing blank sections



Condition

Affected releases: All.

Cause

In some cases, Performance Advisor might be unable to determine the exact cause of the problem, and has assigned a cause of 'unknown' to that region of the chart. Unknown regions might occur if:

- The data from Streamline is too noisy
- Frame markers don't align well with the workload boundaries
- There is a scheduling issue

Workaround

Do the following:

1. Update the firmware on your device to the latest version.
2. Try capturing the profile again with Streamline, using the latest version of Mobile Studio.
3. When generating the report with Performance Advisor, use the `--main-thread` option to explicitly provide the name of the main thread in your application. If there are scheduling issues, Performance Advisor will report it.

Related information

- More FAQs
- Ask a question on the Arm Community forum.

8. Performance Advisor returns an assertions error

When generating a report with Performance Advisor, the report fails and Performance Advisor returns an assertion error: `java.lang.AssertionError`.

Condition

Affected releases: Mobile Studio 2020.0.

Cause

This issue occurs when connecting to an application that is already running. Performance Advisor captures frame annotation events for a number of frames with a negative start time relative to the official start of the Streamline capture. These all get clipped to time=0, which results in an assertion for zero length frames.

Workaround

Start the Streamline capture first, and then start the application. This avoids the frame events with negative start time being generated.

If this is not possible, you can disable assertions in the `pa.ini` file. This file is located in the Arm Mobile Studio installation directory:

```
<install_location>/Arm_Mobile_Studio_2020.0/performance_advisor/pa.ini
```

On macOS, this file is located in:

```
<install_location>/Arm_Mobile_Studio_2020.0/performance_advisor/Performance
Advisor.app/Contents/Eclipse/pa.ini
```

Open this file in a text editor and remove the line `-ea`:

```
-startup
plugins/org.eclipse.equinox.launcher_1.5.500.v20190715-1310.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.gtk.linux.x86_64_1.1.1100.v20190907-0426
--launcher.suppressErrors
--launcher.appendVmargs
-data
@user.home/.pa_workspace
-vm
../java/bin
-vmargs
-ea
-Declipse.exitdata=
-Dstreamline.license.variables=ms
-XX:MaxRAMPercentage=75.0
```

Related information

- More FAQs

- Ask a question on the Arm Community forum.

9. Performance Advisor no longer reports target information in JSON reports

Continuous integration systems set up using Performance Advisor no longer report target information with the JSON key `targetInfo`.

Condition

Affected releases: Mobile Studio 2020.1.

Cause

In Arm Mobile Studio 2020.1, the JSON key `targetInfo` has changed to `deviceInfo`. If you were using the JSON output format in a previous release of Arm Mobile Studio, you will need to update the name in your existing JSON reports and dashboards to maintain compatibility.

Solution

In any JSON reports generated using Arm Mobile Studio 2020.0 or earlier, locate the `targetInfo` key and change it to `deviceInfo`. Check any reporting dashboards that you have created and update any queries that use this key.

Related information

- Performance Advisor
- Export performance data as a JSON file in the Performance Advisor user guide.
- More FAQs
- Ask a question on the Arm Community forum.

10. Performance Advisor reports incorrect frame rate

Reported values for frame rate (FPS) in Performance Advisor and Streamline are incorrect.

Condition

Affects devices running Android 10 or later.

Cause

For devices running Android 9 or earlier, you need to add the Arm interceptor library to your application in order for Arm Mobile Studio tools to collect frame data from it. For devices running Android 10 or later, Arm Mobile Studio tools can use GLES layers or Vulkan validation layers to collect this information from the device. To use this you need to use a command-line option when running the `lwi_me.py` script, to specify the layer library file.

If you have added an Arm interceptor library into your application, and have also specified a layer library at runtime, then the frame rate data reported in Streamline and Performance Advisor may be incorrect. The problem can be confirmed by capturing an Android logcat log, and checking if the log tags `libAGA` and `aga_interceptor` are present. If both tags occur you have loaded both a built-in interceptor and a layer driver, and you may be impacted by this problem.

Solution

On Android 10 you do not need to integrate the built-in interceptor library into the application APK. Instead, you should only use the layer driver framework for both OpenGL ES and Vulkan. Remove the built-in interceptor library from your application and specify the layer library when running the `lwi_me.py` script, using the `--lwi-gles-layer-lib-path` or `--lwi-vk-layer-lib-path` options.

Related information

- Add the Arm interceptor library to your application
- Capture a Streamline profile
- More FAQs
- Ask a question on the Arm Community forum.

11. Performance Advisor fails to capture frame data from Unity applications on Android 9

Performance Advisor fails to generate a performance report, giving the error 'Can't find any frames'.

Condition

Applications built using Unity 2021.2, installed on a device running Android 9.

Cause

Unity 2021.2 removes support for automatically loading the library file `libMGD.so`, which is required for Android 9 or earlier. This file is usually built in to the Unity application, as described in Prepare your Unity project.

Without this file, Performance Advisor can't collect frame data from your application on devices running Android 9.

Workaround

Performance Advisor can install OpenGL ES and Vulkan interceptor layers at runtime, by using the Android OS layer driver mechanisms and the `lwi_me.py` helper script. For OpenGL ES, this OS mechanism is only available on Android 10 and newer. There is currently no fallback for versions of Unity that no longer support loading `libMGD.so` directly.

Either:

- Use an earlier version of Unity, adding the library file as described in Prepare your Unity project
- Upgrade your device to Android 10 or later.

Related information

- More FAQs
- Ask a question on the Arm Community forum.

12. Slow capture with lwi_me.py on Vulkan applications

Slow PA capture on Vulkan apps.

This issue is fixed in Arm Mobile Studio release 2022.0 and later.

Streamline fails to capture data from a Vulkan application when using the `lwi_me.py` script with `--lwi-mode` set to `capture`. Streamline connects to the device and the capture starts, but the frame rate drops to below 10 FPS, and the application on the device runs very slowly, not in real-time as would be expected.

Condition

Vulkan applications, Arm Mobile Studio version 2020.3 and earlier.

Cause

Arm Mobile Studio's lightweight interceptor (LWI) can only capture frames from your application if the interceptor API layer and midstream are active. This causes a problem when capturing data from Vulkan applications. We are working on a fix to this issue, to be included in a future release.

Workaround

If you are not interested in capturing slow frames, run `lwi_me.py` with `--lwi-mode` set to `none`. This will enable you to generate a capture, but will not show slow frame images on the frame analysis chart in Performance Advisor.

If you need to capture slow frames, a workaround is to capture the frame numbers first in 'tag' mode and then capture those frames in 'replay' mode as described in Tagging slow frames in the Performance Advisor user guide.

Related information

- More FAQs
- Ask a question on the Arm Community forum.

13. What is the difference between Graphics Analyzer and Mali Graphics Debugger?

Graphics Analyzer is a rebranded version of Mali Graphics Debugger, and will replace Mali Graphics Debugger for future tool releases. The name was changed to highlight that the tool is no longer restricted to supporting platforms only running on a Mali GPU, and will now enable debug of graphical applications irrespective of the GPU in the target device.

Download Graphics Analyzer

Graphics Analyzer is included with Arm Mobile Studio and Arm Development Studio.

For graphics and gaming developers, download Arm Mobile Studio to get Graphics Analyzer with a free starter edition license, and start using it today.

For embedded C/C++ software developers, try Arm Development Studio for free for 30 days.

Related information

- More FAQs
- Ask a question on the Arm Community forum.

14. Graphics Analyzer Device Manager fails to connect to my device

Graphics Analyzer's Device Manager fails to connect to the device either from the devices list, or by using the Connect to an IP fields.

Condition

Affected releases: Mobile Studio 2019.1, 2019.2, 2020.0, 2020.1.

Cause

In versions of Arm Mobile Studio prior to 2020.2, Graphics Analyzer requires you to use a Python script to connect to your device.

Solution

Do the following:

1. In a terminal, navigate to the Arm Mobile Studio installation directory, where you will find a Python script, named `aga_me.py` which helps Graphics Analyzer capture data from your device:

   ```
   cd <install_directory>/graphics_analyzer/target/android/arm/
   ```

2. Run the `aga_me.py` script, giving the Android package name of the application you want to profile.

   If you don't know the package name, run the script with the `--list_packages` option first, to see a list of the debuggable packages installed on the device:

   ```
   python3 aga_me.py --list_packages
   ```

   For OpenGL ES applications, run the script with:

   ```
   python3 aga_me.py com.MyCompany.MyApp
   ```

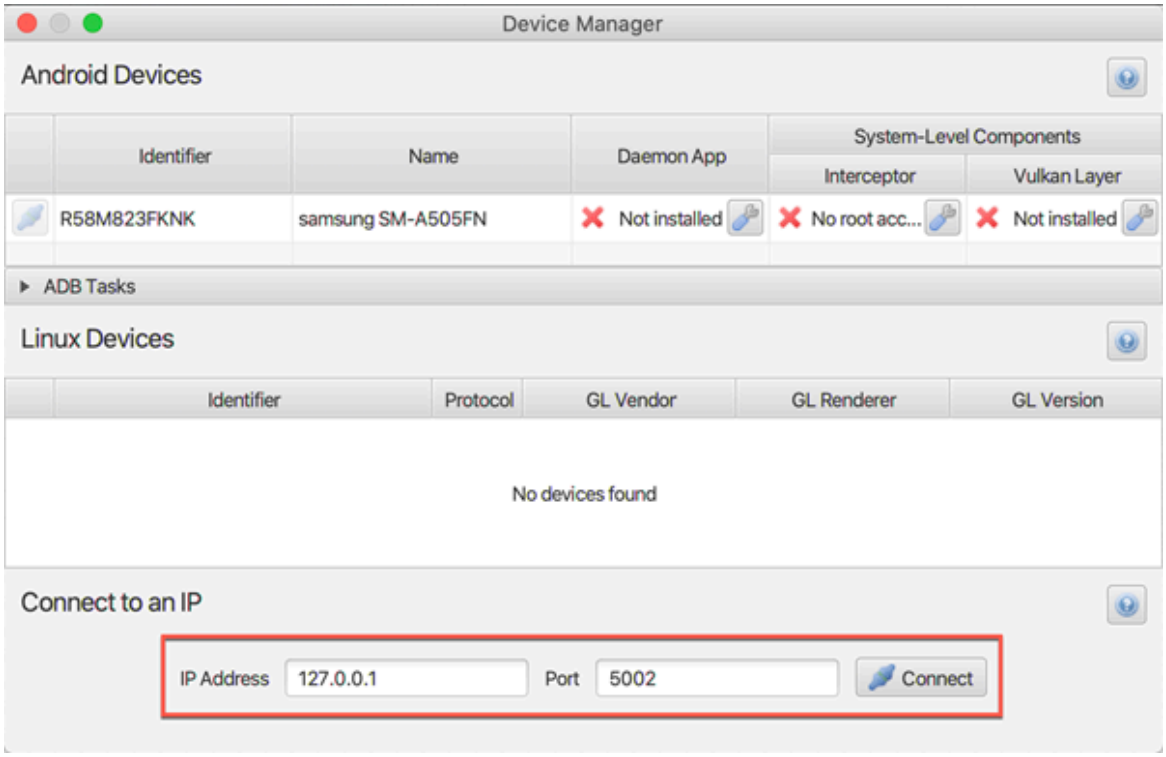
   For Vulkan applications, use the `-v` option to instruct the script to install the Vulkan layer. Refer to the Android documentation for more details.

   ```
   python3 aga_me.py -v com.MyCompany.MyApp
   ```

3. Launch Graphics Analyzer and select Open the Device Manager from the Debug menu. If your device is connected successfully it will be listed in the Android Devices window. However, don't use the button here to make the connection, as this may not work for some Android versions. Instead, use the Connect to an IP fields at the bottom of the dialog.
   a. Specify an IP Address of 127.0.0.1
   b. Specify the Port number 5002

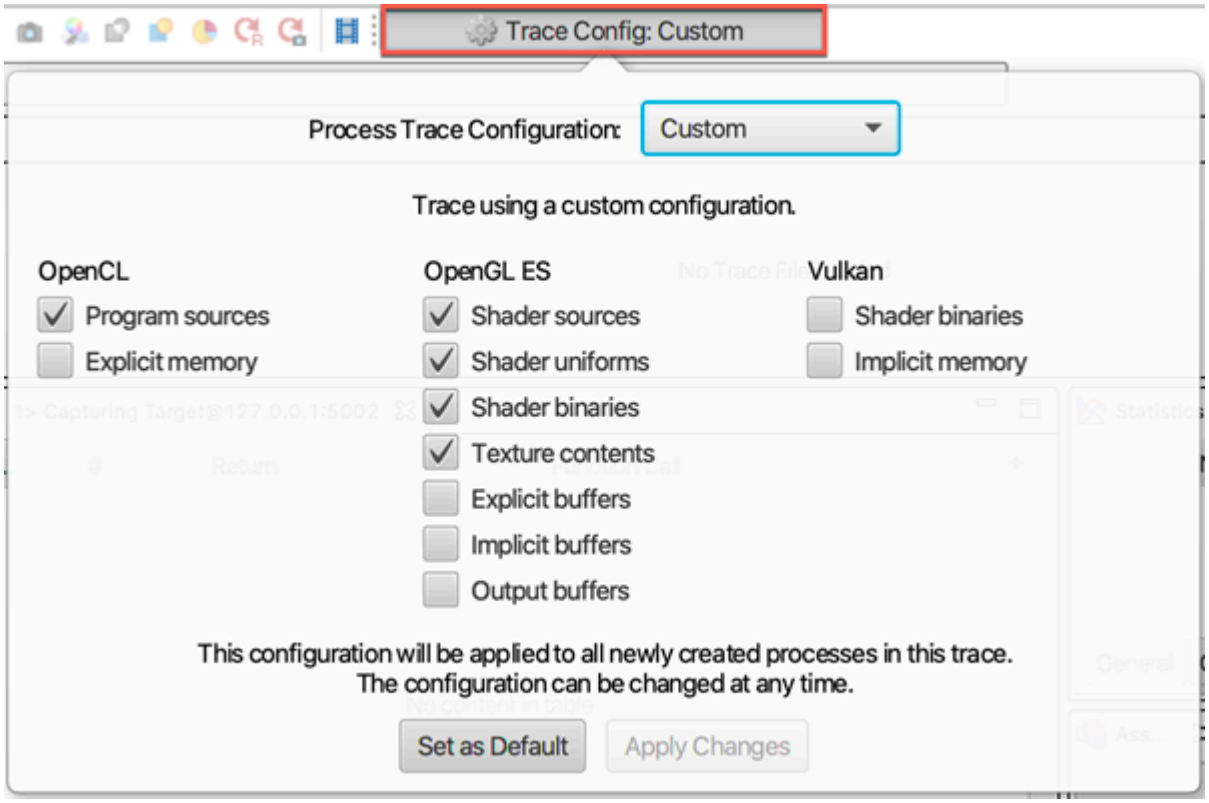
c. Click Connect.

Figure 14-1: Graphics Analyzer Device Manager



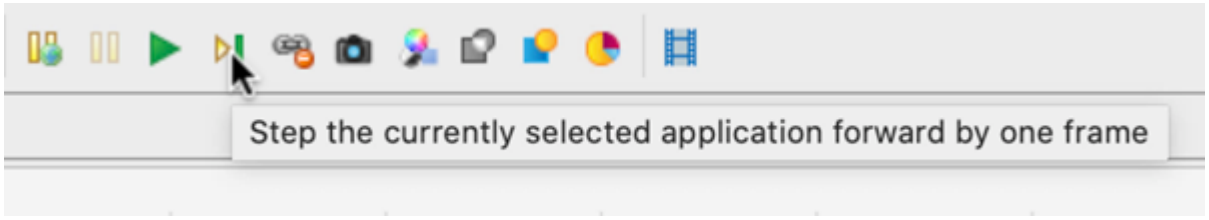
4. Optionally, select Trace Config and select which API assets are captured. Only enable the asset types you need. The more asset types you enable, the slower the application will run, the more memory is required, and the generated trace file will be larger.

Figure 14-2: Graphics Analyzer Trace Configuration



5. In the terminal, press Enter to continue the script. The script will launch your application on the device and start capturing the trace. Graphics Analyzer begins displaying the trace data as it receives it from the target.
6. Perform your test scenario on the device. When you get to a problem area, use the pause, step and play buttons to locate a frame that you want to analyze more closely:

Figure 14-3: Graphics Analyzer buttons

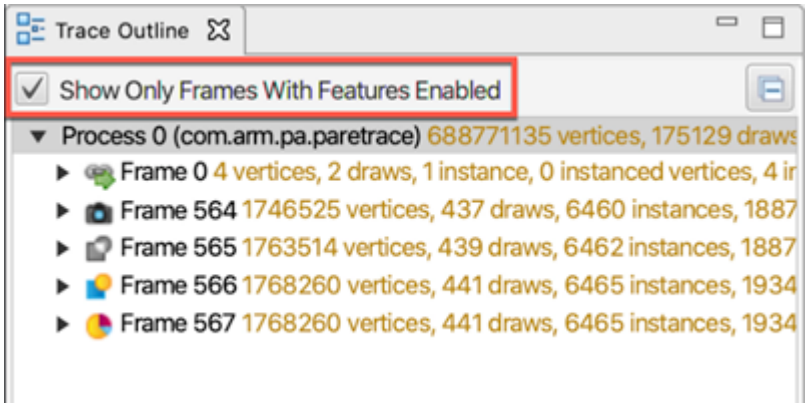


7. Click the camera icon to capture the frame buffer output at the current frame.
8. Optionally, capture extra frame data by enabling the following modes:
   a. Overdraw
   b. Shader map
   c. Fragment count

Enable the mode, then click the camera icon to collect the data.

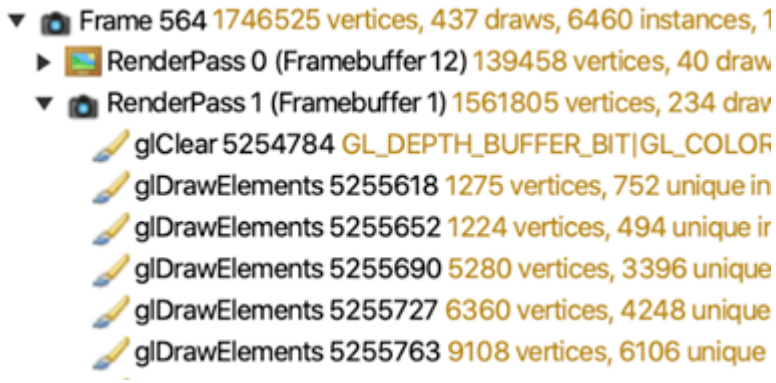
9. To stop tracing, press Enter in the terminal to continue the script. The script will stop the trace, remove the Graphics Analyzer daemon from your device, and terminate. Graphics Analyzer will report a warning that connection to your device has been interrupted. You can safely ignore this warning.
10. Analyze your trace using the various data views in Graphics Analyzer. All the frames are listed in the Trace Outline view. The frames where you've captured extra data are shown with an icon, to identify the type of frame capture you performed.
    a. To filter the frames to just those where you've captured extra data, use the Show Only Frames With Features Enabled option

Figure 14-4: Filtering the trace outline in Graphics Analyzer



    b. Expand a frame to see the renderpasses and draw calls within it.

Figure 14-5: Expanding frames in Graphics Analyzer



    c. Select frames, renderpasses and draw calls to explore their data. Refer to the Graphics Analyzer user guide information about the different data views.
11. Save or export the trace file, using options under the File menu.

15. Graphics Analyzer becomes unresponsive on closing

Graphics Analyzer does not close and becomes unresponsive.

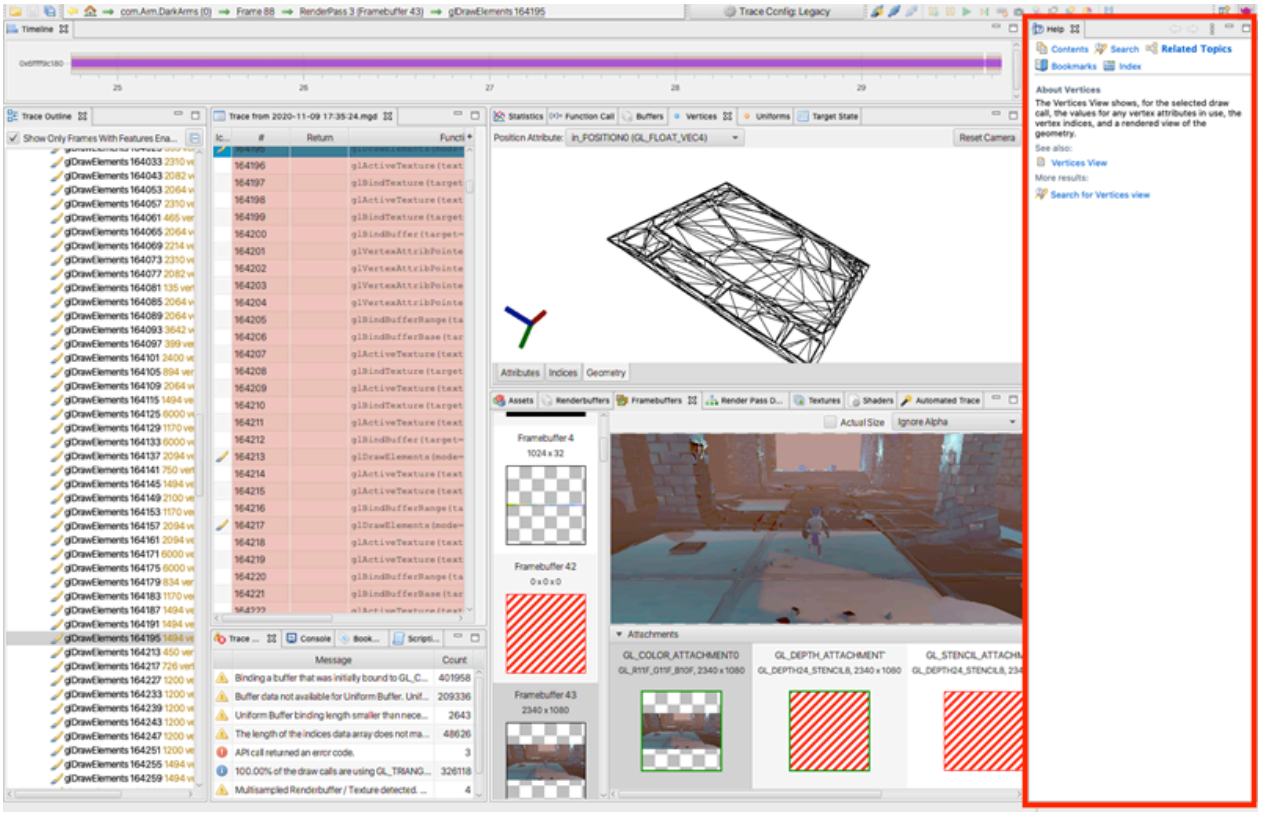
Condition

Affected platforms: Linux.

Cause

Linux platforms, if you try to close Graphics Analyzer while the dynamic help view is open, Graphics Analyzer does not close and becomes unresponsive.

Figure 15-1: Graphics Analyzer with dynamic help active



Workaround

To avoid this issue, ensure that you close the dynamic help view before closing Graphics Analyzer. Then you can close the application normally.

Related information

- More FAQs

- Ask a question on the Arm Community forum.

16. Graphics Analyzer fails with a socket permissions error

When running Graphics Analyzer on an Android target, connection fails and Graphics Analyzer returns a socket permissions error.

Condition

Affected releases: Mobile Studio 2019.0, Mobile Studio 2019.1.

Fixed in Mobile Studio 2020.0.

Cause

The Graphics Analyzer interceptor uses a local domain socket to connect the application being instrumented to the on-target daemon, which is used to package the data for the host machine. In recent versions of the Android SDK, applications are placed inside a more restrictive security sandbox which blocks access to the method we use to create the local domain socket.

Workaround

In later versions of Arm Mobile Studio, this process is much easier. You should download and install the latest version of Arm Mobile Studio, and follow the getting started tutorial for instructions on how to connect.

To avoid this issue in earlier versions, build the application being tested using a manifest, setting the `targetSdkVersion` to 25 or lower.

Related information

- More FAQs
- Ask a question on the Arm Community forum.

17. Graphics Analyzer fails to capture data from my Unity app

In applications built with Unity, Graphics Analyzer fails to capture a trace even though the device is connected and the interceptor library is present and set correctly in the Unity project.

Condition

Affected releases: Mobile Studio 2019.0, Mobile Studio 2019.1, Mobile Studio 2019.2.

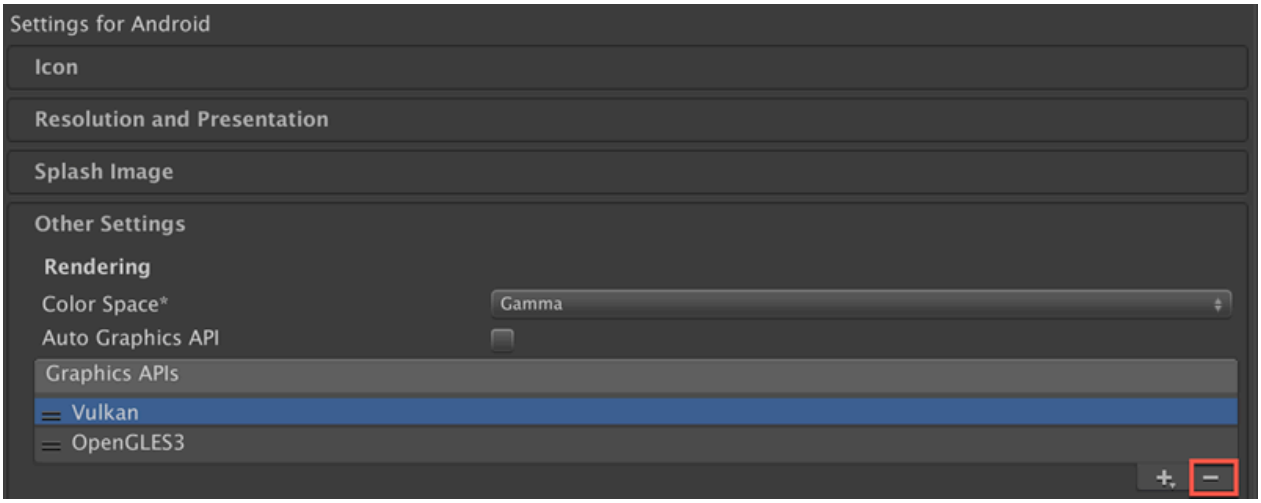
Cause

Unity enables the Vulkan API and OpenGLES3 graphics APIs by default. Building for the Vulkan API prevents Graphics Analyzer from intercepting successfully in the impacted releases.

Workaround

Select File > Build Settings and select Player Settings. Under Rendering, remove Vulkan from the list of Graphics APIs.

Figure 17-1: Changing Graphics API settings in Unity



Related information

- More FAQs
- Ask a question on the Arm Community forum.
- Arm guide for Unity developers

18. Graphics Analyzer host Device Manager can not find the daemon APK

In Arm Mobile Studio 2019.0, you could use the Device Manager to install Graphics Analyzer's daemon APK on a connected Android device. In some cases this fails to work correctly.

Condition

Affected releases: Mobile Studio 2019.0.

Fixed in Mobile Studio 2019.1.

Cause

Arm Mobile Studio 2019.0 used a signed macOS binary for the application but does not use a signed disk image for the data files the application uses, which means that the application cannot access the APK to install it.

Workaround

Device connection is handled using a connection script in later versions of Arm Mobile Studio, which makes this process much easier. You should download and install the latest version of Arm Mobile Studio, and follow the getting started tutorial for instructions on how to connect.

To avoid this issue in 2019.0, manually install the APK using `adb install` from the command line. The file is located in the installation directory:

```
adb install <install_location>/graphics_analyzer/target/android/arm/AGA.apk
```

Related information

- Get started with Graphics Analyzer
- More FAQs
- Ask a question on the Arm Community forum.

19. Graphics Analyzer fails when tracing Android 10 applications

Graphics Analyzer fails when intercepting API calls from applications that contain native C++ code, on devices running Android 10(Q). The interceptor starts normally, but shuts down after the application starts, capturing only one API call. There are no error messages.

Condition

Affected releases: Mobile Studio 2019.0, Mobile Studio 2019.1, Mobile Studio 2019.2.

Cause

This problem occurs because there are changes to the bionic libraries and dynamic linker paths in Android 10.

Workaround

To resolve this issue, you must move the `System.loadLibrary()` call from the start of the main activity to inside the Native C++ activity constructor. Call the load method after the `super.onCreate()` call in the `onCreate` method:

```
public class YourNativeActivity extends NativeActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // Call super-class onCreate first
        super.onCreate(savedInstanceState);

        // Load interceptor second
        try {
            System.loadLibrary("AGA");
        } catch (UnsatisfiedLinkError e) {
            e.printStackTrace();
        }
        ...
    }
}
```

Related information

- More FAQs
- Ask a question on the Arm Community forum.

20. Graphics Analyzer can not access Khronos reference pages

In Graphics Analyzer, double-clicking a function does not open the Khronos reference page for that function, as expected.

Condition

Affected platforms: Windows, Internet Explorer.

Cause

Some browsers are not compatible with the Khronos reference page format. In particular, some versions of Internet Explorer do not support this functionality.

Workaround

Try installing a different browser and setting it to the default browser on your Windows machine.

Related information

- More FAQs
- Ask a question on the Arm Community forum.

21. Graphics Analyzer playback and capture buttons unavailable

In some cases, when capturing a trace of an Unreal Engine application using Graphics Analyzer, the playback and capture buttons are not available and appear grayed-out.

Figure 21-1: Capture buttons are unavailable in Graphics Analyzer

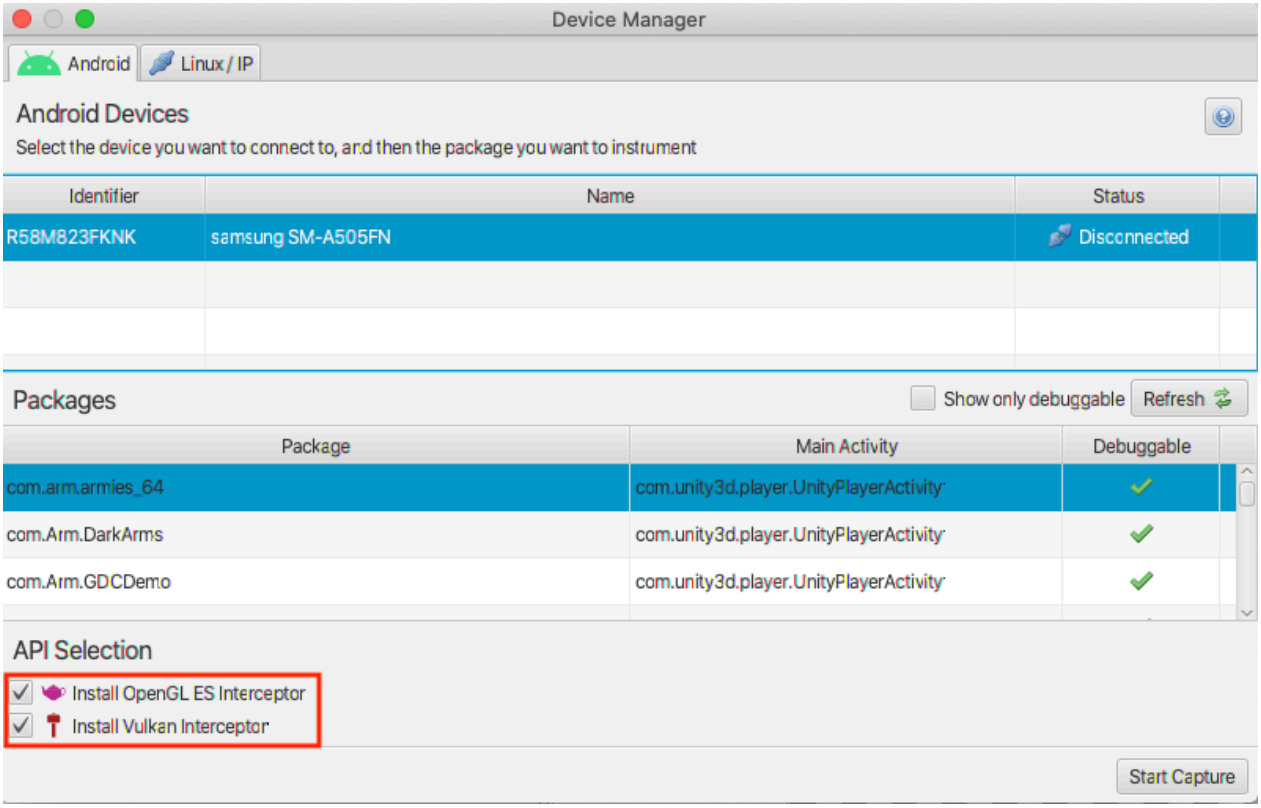
Condition

Applies to all versions of Graphics Analyzer when tracing Unreal Engine applications that use mixed Vulkan and OpenGL ES.

Cause

This problem can occur if both GLES and Vulkan APIs are selected when starting the capture.

Figure 21-2: Device Manager in Graphics Analyzer with multiple APIs selected



Workaround

The following workarounds apply:

- If your application is OpenGL ES only, deselect the option to Install Vulkan Interceptor from the Device Manager when setting up the capture
- If you need to capture both OpenGL ES and Vulkan API calls, you will need to select both interceptors on the Device Manager. In this case, use the global pause button to pause the application as soon as tracing starts, and then press play to resume. The capture buttons should now be available.

Related information

- More FAQs
- Ask a question on the Arm Community forum.

22. Graphics Analyzer fails to open on RHEL 8 CentOS 8

On CentOS 8 systems, Graphics Analyzer fails to launch and reports the following error:

```
Gdk-CRITICAL **: 12:35:22.301: gdk_x11_display_set_window_scale: assertion
 'GDK_IS_X11_DISPLAY (display)' failed
```

Condition

Arm Development Studio installed on RHEL 8 CentOS 8 systems.

Cause

A known issue related to the backend code that JavaFX uses is causing the application to crash.

Workaround

Set the environment variable `GDK_BACKEND` to `x11`:

```
export GDK_BACKEND=x11
```

Related information

- More FAQs
- Ask a question on the Arm Community forum.

23. Mali Offline Compiler fails to start on Windows

When launching Mali Offline Compiler on Windows platforms, the application fails to start, and issues the following error message:

'The code execution cannot proceed because `VCRUNTIME140_1.dll` was not found. Reinstalling the program may fix this problem.'

Figure 23-1: Error message

Condition

Affected releases: Mobile Studio 2020.0.

Affects Windows platforms.

Cause

A required `.dll` file is missing from your system.

Workaround

To fix this problem, reinstall the latest 64-bit redistributable for Visual Studio 2015, 2017 and 2019 from the Microsoft support page.

Related information

- Get started with Mali Offline Compiler
- More FAQs
- Ask a question on the Arm Community forum.

24. Security warning when starting Mali Offline Compiler on macOS

When launching Mali Offline Compiler on macOS 10.15 (Catalina), the application fails to start, and issues the following error message:

‘“malioc” cannot be opened because the developer cannot be verified’.

Condition

Affected releases: Mobile Studio (all versions).

Affects macOS 10.15 (Catalina) platforms.

Cause

macOS Catalina introduced changes to its Gatekeeper functionality. Because the Mali Offline Compiler binary is not in the ‘.app’ format, macOS Catalina issues a security warning. You can verify the integrity of Mali Offline Compiler’s binary by entering the following command in a Terminal window:

```
spctl -a -vv malioc
malioc: rejected (the code is valid, but does not seem to be an app)
origin=Developer ID Application: ARM Ltd (S345482SL3)
```

Solution

Gatekeeper checks can be disabled by manually removing the `com.apple.quarantine` flag from the installed Mali Offline Compiler binary. Open a Terminal window, and run the following commands to disable gatekeeper:

1. Navigate to the Mali Offline Compiler installation directory:

   ```
   cd /Applications/Arm_Mobile_Studio_2020.0/mali_offline_compiler/
   ```

2. Enter the following command:

   ```
   sudo xattr -d com.apple.quarantine malioc
   ```

Related information

- Get started with Mali Offline Compiler
- More FAQs
- Ask a question on the Arm Community forum.

25. Related information

Here are some resources related to material in this guide:

- Ask a question
- Documentation
- Supported devices