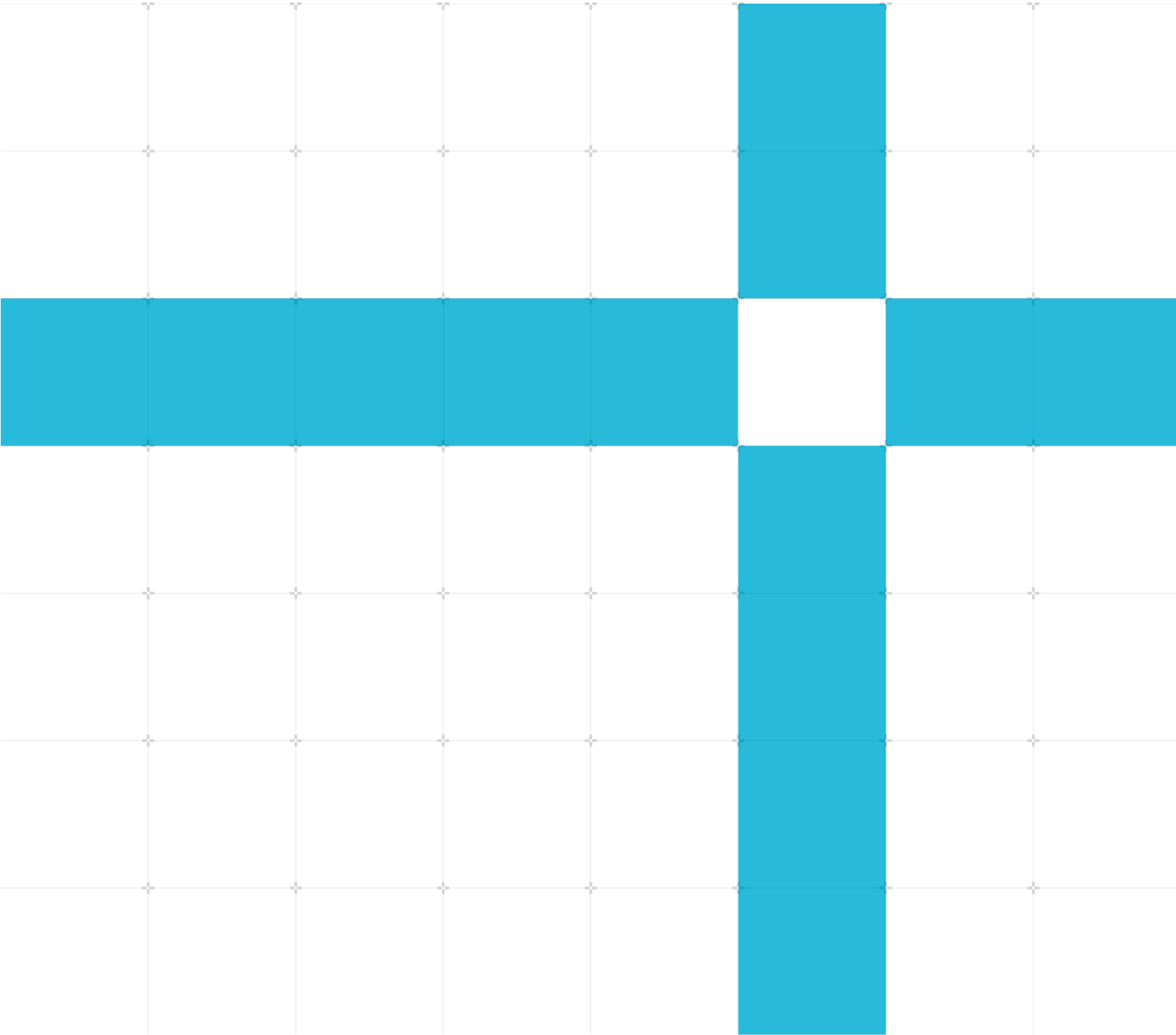
Arm® Mobile Studio 2022.4

Product revision: r22p4-00rel0

Release Note

Non-Confidential

Issue 00
DSHVE-DC-06002



Arm Mobile Studio 2022.4

Release Note

Non-Confidential Proprietary Notice

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.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on Arm Mobile Studio, 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.

To report offensive language in this document, email terms@arm.com.

Contents

1 Release overview 5
1.1 Product description 5
1.1.1 Component versions 5
1.2 Release status 5
1.3 Changes in this release 5
1.3.1 Mobile Studio 5
1.3.2 Streamline 6
1.3.3 Performance Advisor 6
1.3.4 Graphics Analyzer 7
1.3.5 Mali Offline Compiler 7
1.3.6 Mobile Studio for Unity package 7
1.4 Known issues in this release 8
1.4.1 Streamline 8
1.4.2 Performance Advisor 8

2 Support 9
2.1 How-to videos 9
2.2 Host OS support 9
2.3 Target OS support 9
2.4 The Mobile Studio for Unity package 10

3 Installation 11
3.1 Install on Windows 11
3.2 Install on macOS 11
3.3 Install on Linux 12

1 Release overview

The following sections describe the product and its quality status at time of release.

1.1 Product description

Arm® Mobile Studio is a tool suite enabling Android application developers to detect performance bottlenecks in their Arm CPU software and Arm Mali™ GPU rendering. Profiling is provided through analysis of performance counters from the hardware, and the target application's graphics API usage.

This release of Arm Mobile Studio includes:

- Streamline, for profiling application software and rendering performance.
- Performance Advisor, for automating initial data analysis and reporting in continuous integration deployments.
- Graphics Analyzer, for debugging and inspecting usage of graphics APIs.
- Mali Offline Compiler, for static analysis of shader programs and compute kernels.

1.1.1 Component versions

This release of Arm Mobile Studio includes the following tool versions:

- Streamline 8.3
- Performance Advisor 8.3
- Graphics Analyzer 5.12
- Mali Offline Compiler 7.8

1.2 Release status

This is the REL quality release of the Arm Mobile Studio 2022.4 (r22p4-00rel0) software.

1.3 Changes in this release

This release of Arm Mobile Studio contains the following changes.

1.3.1 Mobile Studio

Mobile Studio has the following product-wide changes:

- Mobile Studio is now only available as a single free-of-charge edition. All licensed features from the Professional Edition are now available for all users without needing a license:

- Capture and analyze data with headless continuous integration deployments, to automate performance monitoring throughout the development cycle.
- Export machine-readable reports in CSV and JSON format for use in custom data analysis.
- Profile any application on Android Eng or UserDebug builds.

1.3.2 Streamline

Streamline has the following changes:

- Arm Immortalis ™ -G715, Mali-G715, and Mali-G615 are supported as profiling targets.
- Software profiling now supports application binaries using the DWARF5 debug format.
- Analysis time and memory footprint for software profiles containing a large amount of application debug info has been significantly improved. Time to analyze a sample Unreal Engine project with ~3GB of debug info dropped from 25 minutes to 2.5 minutes.
- Fix: Data parsing correctly handles out-of-order annotation packets, which could cause software counters and event annotations to be incorrectly discarded during analysis.
- Fix: An intermittent gatord crash that terminated captures too early has been fixed. This crash would manifest as an intermittent segmentation fault in the “gatord-iocx-0” child process.

1.3.3 Performance Advisor

Performance Advisor report generation has the following changes:

- Immortalis-G715, Mali-G715, and Mali-G615 are supported as profiling targets.
- Report generation now supports the following new options:
  - `--region-report-min-length=length`
    Short regions below a given minimum length are omitted from reports.
  - `--region-report-max-depth=level`
    Regions deeper than a given level in the region hierarchy are omitted from reports.
- Report generation is now more stable for multi-context OpenGL ES content, or multi-device Vulkan content. The new default behavior is to select the context containing the most frames, but a context can also be manually selected if needed.
- Fix: Report charts now correctly render Streamline counters that are using the “Maximum” aggregation type, such as software counters, and proc memory statistics.

Performance Advisor’s light-weight interceptor and support script have the following changes:

- The interceptors for both OpenGL ES and Vulkan have been replaced by a new implementation. They improve reliability, especially for Vulkan, and reduce the performance impact on the target application.
- The OpenGL ES interceptor is now only provided as a layer driver, which requires Android 10 or newer. Applications on older devices can still use Performance Advisor by manually emitting the required frame boundary annotations from the application.

- OpenGL ES slow-frame screen capture has been changed to skip screenshots if the previous screenshot is still being saved. This removes performance hitches caused by a backlog of screenshots building up in the application.
- OpenGL ES slow-frame screen capture has been changed to write **.bmp** images rather than uncompressed **.png** images when running in uncompressed mode. This reduces the time to capture and write a screenshot from 250ms to under 80ms, reducing the performance impact on the application.
- The following **lwi_me.py** script options have been removed:
    - `--32-bit`
      The application bitness is now auto-detected.
    - `--lwi-mode=alone`
      This option is no longer useful.
- **Fix:** OpenGL ES slow-frame screen captures in multi-context applications are now correctly namespaced with the EGLContext that created them.
- **Fix:** OpenGL ES slow-frame screen captures in a timed headless capture now cease when the headless timeout is reached, avoiding a stability issue caused by screenshots being written faster than the data could be read from the device.
- **Fix:** Vulkan interceptor no longer causes validation layer failures.

1.3.4 Graphics Analyzer

Graphics Analyzer has the following changes:

- Device configuration now only allows either OpenGL ES or Vulkan to be instrumented. It is no longer possible to inject layers for both APIs for a single capture.
- **Fix:** Vulkan interceptor no longer tries to query surface properties from an optimal layout surface. This fixes crashes and validation failures on the latest Pixel 6 OS update.

1.3.5 Mali Offline Compiler

Mali Offline Compiler has the following changes:

- Compiler backend for Bifrost and Valhall architecture GPUs has been updated to the r41p0 driver.
- Vulkan ray pipeline shader stages can now be compiled for the Immortalis-G715, Mali-G715, and Mali-G615 GPUs.
- Vulkan ray query best practice guidelines that must be followed to avoid the slow traversal path have been updated in the User Guide.
- The load/store unit cost model for Bifrost and Valhall architecture GPUs has been improved, and now correctly reflects the lower access cost for uniform loads and stack access.

1.3.6 Mobile Studio for Unity package

The Mobile Studio for Unity package (available on GitHub) contains the following changes:

- C# bindings for CAM annotations now support nesting tracks hierarchically.
- C# bindings for CAM annotations now allow dependencies between Jobs to be expressed.

- C# binding for counter `set_value()` renamed to `setValue()`.

1.4 Known issues in this release

This release of Arm Mobile Studio contains the following known issues.

1.4.1 Streamline

Streamline has the following known issues:

- SDDAP-11607: macOS host installs do not show Arm disassembly views.
- SDDAP-11426: Linux host installs using NVIDIA drivers can experience areas of the UI rendering as black rectangles when using monitor scaling. This can be worked around by setting the environment variable GDK_SCALE to 1 before launching Streamline. For ease of use, this can be set in the Streamline launch script.
- SDDAP-11768: CAM annotations ignore user-specified colors and always use the default color sequence.

1.4.2 Performance Advisor

Performance Advisor has the following known issues:

- SDDAP-11717: Vulkan screenshots are not currently supported.

2 Support

To help you get started we provide a number of quick start guides available online:

- Getting Started Guides on developer.arm.com

Technical support for Arm Mobile Studio is provided via our developer forums:

- Developer forums on community.arm.com

2.1 How-to videos

Refer to the following videos to learn how to use Arm Mobile Studio tools.

- How to capture a performance profile of your application with Streamline
- How to capture a trace of your application with Graphics Analyzer
- How to generate a report with Performance Advisor
- How to analyze a shader program with Mali Offline Compiler

To learn more about Mali GPUs and how to develop optimized graphics content for mobile devices, refer to our Mali GPU Training Series.

2.2 Host OS support

This release has been developed for the following host operating systems:

Table 2-1: Host operating system used in developing this release

Operating system	Version
Windows	10 or newer
macOS	10.13 (High Sierra) or newer In this release support for macOS 10.13 and 10.14 is deprecated. Mobile Studio 2023.0 will require macOS 10.15 (Catalina) or newer.
Ubuntu Linux	18.04 (Bionic Beaver) or newer

2.3 Target OS support

This release has been developed for the following target operating systems:

Table 2-2: Target operating system used in developing this release

Feature	Version
Streamline	Android 8 or newer
Performance Advisor OpenGL ES	Android 8 or newer with manual annotation Android 10 or newer to use the Light-weight Interceptor
Performance Advisor Vulkan	Android 9 or newer
Graphics Analyzer OpenGL ES	Android 8 or newer

Graphics Analyzer Vulkan	Android 9 or newer

2.4 The Mobile Studio for Unity package

The Mobile Studio for Unity package provides a supporting Unity game engine integration for Streamline and Performance Advisor. The package provides:

- C# bindings for Streamline's annotation API, allowing users to export custom software counters, and event annotations.
- Integration with the Unity profiler data source, exporting Unity object counts and memory allocations as custom software counters.

The annotation API provides a generic means to markup a Streamline capture. It can be used to emit the semantic tags that Performance Advisor reports use to denote interesting gameplay regions.

The package is available on GitHub, and can be imported directly into your Unity project using the Unity package manager. See the GitHub project documentation for more details.

- https://github.com/ARM-software/mobile-studio-integration-for-unity/

3 Installation

This guide describes how to install and configure Arm Mobile Studio to run on 64-bit Windows, macOS®, and Linux.

Mobile Studio requires **Android Debug Bridge (ADB)** and **Python 3.5** (or newer), to enable connection to your device. Make sure you have **these tools** installed and that you have configured your environment to use them.

3.1 Install on Windows

Arm Mobile Studio is provided with an installer executable. Double-click the **.exe** file and follow the instructions in the setup wizard.

- To launch Streamline, open the Windows Start menu, navigate to the Arm Mobile Studio folder, and select the "Arm MS Streamline 2022.4" shortcut,
- To launch Graphics Analyzer, open the Windows Start menu, navigate to the Arm Mobile Studio folder, and select the "Arm MS Graphics Analyzer 2022.4" shortcut.
- To launch Performance Advisor, open a command terminal, navigate to your work directory, and run the `pa` command on a Streamline capture file. The `pa` command is added to your PATH during installation, so can be used from anywhere.
  ```
  pa.exe <options> my_capture.apc
  ```
- To launch Mali Offline Compiler, open a command terminal, navigate to your work directory, and run the `malioc` command on a shader program. The `malioc` command is added to your PATH during installation, so can be used from anywhere.
  ```
  malioc.exe <options> my_shader.frag
  ```

3.2 Install on macOS

Arm Mobile Studio is provided as a **.dmg** package. To mount it, double-click the **.dmg** package and follow the instructions. The Mobile Studio directory tree is copied to the **Applications** directory on your local file system for easy access.

Launch the tools directly from the Arm Mobile Studio directory tree in your Applications directory.

- To launch Streamline, go to the **<installation>/streamline** directory, and open the **Streamline.app** file.
- To launch Graphics Analyzer, go to the **<installation>/graphics_analyzer/gui** directory and open the **Graphics Analyzer.app** file.
- To launch Performance Advisor, go to the **<installation>/performance_advisor** directory, and double-click the **performance_advisor_launcher** file.

  Your computer will ask you to allow Performance Advisor to control the Terminal application. Confirm this.

  The Performance Advisor launcher opens the Terminal application and updates your PATH environment variable so you can run the `pa` command from any directory.

Run the `pa` command on a Streamline capture file to generate a report:

```
pa <options> my_capture.apc
```

- To launch Mali Offline Compiler, go to the **<installation>/mali_offline_compiler** directory, and double-click the **mali_offline_compiler_launcher** file.

  The Mali Offline Compiler launcher opens the Terminal application and updates your PATH environment variable so you can run the `malioc` command from any directory.

  Run the `malioc` command on a shader program.

  ```
  malioc <options> my_shader.frag
  ```

On some versions of macOS, you might see a message that Mali Offline Compiler is not recognized as an application from an identified developer. To enable Mali Offline Compiler, cancel this message, then open **System Preferences > Security and Privacy**, and select **Allow Anyway** for the `malioc` application.

3.3 Install on Linux

Arm Mobile Studio is provided as a gzipped tar archive. Extract this tar archive to your preferred location, using a recent version (1.13 or later) of GNU tar:

```
tar xvzf Arm_Mobile_Studio_2020.0_linux.tgz
```

Launch the tools directly from the location where you extracted the package.

- To launch Streamline, go to the **<installation_directory>/streamline** directory and run the **`Streamline`** file.

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

- To launch Graphics Analyzer, go to the **<installation>/graphics_analyzer/gui** directory and run the **aga** file.

  ```
  cd <install>/graphics_analyzer/gui
  ./aga
  ```

- To launch Performance Advisor, go to the **<installation>/performance_advisor** directory and run the `pa` command on a Streamline capture file.

  ```
  cd <install>/performance_advisor
  ./pa <options> my_capture.apc
  ```

- To launch Mali Offline Compiler, go to the **<installation>/mali_offline_compiler** directory and run the `malioc` command on a shader program.

  ```
  cd <install>/mali_offline_compiler
  ./malioc <options> my_shader.frag
  ```