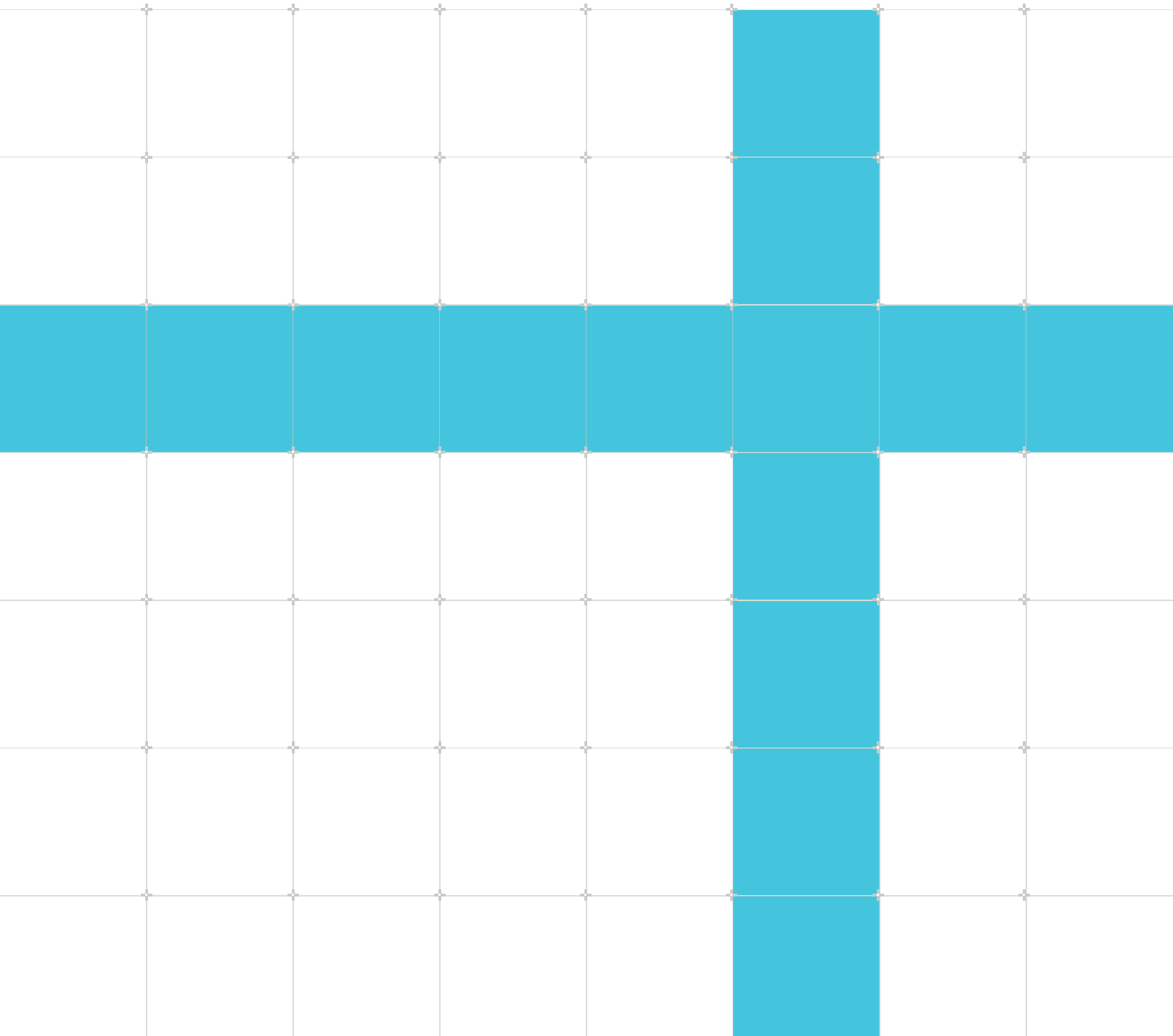
RenderDoc for Arm® GPUs

Version 2025.2

User Guide

Issue 00
109669_2025.2_00_en



RenderDoc for Arm® GPUs User Guide

This document (109669_2025.2_00_en) was issued on 2025-05-01. There might be a later issue at https://developer.arm.com/documentation/109669

The product version is 2025.2.

See also: Proprietary notice | Product and document information | Useful resources

Start reading

If you prefer, you can skip to the start of the content.

Intended audience

This document is intended for software developers who want to use RenderDoc for Arm GPUs for frame-based graphical analysis of Android and Linux applications.

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.

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.

Contents

1. Introduction to RenderDoc for Arm® GPUs

RenderDoc for Arm® GPUs enables RenderDoc users to capture, analyze, and debug graphics applications from Windows, Linux, and Android targets. This section introduces RenderDoc for Arm® GPUs and describes how it works with RenderDoc.

Overview of RenderDoc for Arm® GPUs

RenderDoc for Arm® GPUs is an Arm fork of the RenderDoc open-source graphics API debugger. The Arm release includes support for API features and extensions that are available on the latest Arm GPUs, but are not yet supported in upstream RenderDoc.

Arm contributes changes to the upstream project, but some Arm-specific or Android-specific features are only available in the Arm fork.

Arm-specific features

RenderDoc for Arm® GPUs is based on upstream RenderDoc 1.37 and has the following extensions:

- Bundled pre-built glibc and musl binaries for remote Linux.
- Binary RenderDoc releases for developers using macOS host machines.
- Use `malioc` as a shader view tool.
- Support for capture and replay of Vulkan opacity micromaps.
- Support for libGPUCounters.
- Automatic attachment and swapchain image rotation based on swapchain pre-rotate.
- Android 10 or later, configure Vulkan debug layers to use during capture and replay.
- Open captures remotely on Linux and Android targets.

Support

The following resources provide additional help and information:

Resource	Link
RenderDoc documentation	https://renderdoc.org/docs/index.html
To ask a question directly, you can email the Arm® Performance Studio team	performancestudio@arm.com
RenderDoc for Arm® GPUs on Arm Community	Graphics, Gaming, and VR community forum

Note

The latest version of the RenderDoc for Arm® GPUs User Guide is included in each product release. The user guide version might be older than the product version because the user guide is updated only as required.

2. Get started with RenderDoc for Arm® GPUs

This tutorial describes how to set up a target and capture frames for analysis with RenderDoc for Arm® GPUs.

Learn how to:

- Setup your Android target to prepare your computer and Android target.
- Setup your Linux target to prepare your computer and Linux target
- Capture frames from your application with a mobile application running on your Android target.
- Analyze and debug your capture highlights some of the analysis and debug features that are available in RenderDoc for Arm® GPUs.

Additional learning resources

Additional tutorials and help articles are available as part of the RenderDoc documentation, including:

- How do I use RenderDoc on Android?
- RenderDoc Quick Start
- How-to topics in the RenderDoc documentation

2.1 Setup your Android target

Complete the required setup tasks before you use an Android target with RenderDoc for Arm® GPUs.

Before you begin

Before you can setup your Android target, you must complete the following tasks:

- Install Android Debug Bridge. `adb` is available with the Android SDK platform tools, which are installed as part of Android Studio. Alternatively, you can download them separately as part of the Android SDK platform tools.
- Download Arm Performance Studio for free and follow the installation instructions in the Arm® Performance Studio Release Notes.

Note

Your Android target must be running Android 9.0 or later.

Procedure

1. Connect your target to your computer through USB and ensure that the target is switched on.
2. Enable Developer Mode on your target.
3. On your target, go to **Settings > Developer Options** and enable **USB Debugging**. If your target asks you to authorize connection to your computer, confirm the connection.
   Test the connection by entering `adb devices` in a command-line utility. If successful, the command returns the target ID.

   ```
   adb devices
   List of devices attached
   ce12345abcdf1a1234      device
   ```

   If you see that the target is listed as unauthorized, try disabling and re-enabling **USB Debugging** on the target, and accept the authorization prompt to enable connection to the computer.
4. If you have Android Studio open, it interferes with RenderDoc debugging by attaching to the package itself. You can either close Android Studio, or disable adb integration in Android Studio using the **Tools > Android > Enable ADB integration** setting.

Next steps

After connecting and configuring your Android target, you can now perform an on-target capture using RenderDoc for Arm® GPUs.

- Capture frames from your application

2.2 Setup your Linux target

Complete the required setup tasks before you use a Linux target with RenderDoc for Arm® GPUs.

Before you begin

Before you can setup your Linux target, you must complete the following tasks:

- Download Arm Performance Studio for free and follow the installation instructions in the Arm® Performance Studio Release Notes.
- Ensure that your host is running the minimum required version of Linux, or later:
  - `glibc`: Ubuntu 20.04
  - `musl`: Alpine 3.20.1

Procedure

1. Connect your target to your computer and ensure that the target is switched on.
2. Send the `glibc` and `musl` library files to the server using SCP. You can use any of the following commands to copy the files:

- `glibc`

```
scp share/renderdoc/plugins/aarch64/glibc/bin/renderdoccmd <remote
 device>:<remote path>
scp share/renderdoc/plugins/aarch64/glibc/lib/librenderdoc.so <remote
 device>:<remote path>
```

- `musl`

```
scp share/renderdoc/plugins/aarch64/musl/bin/renderdoccmd <remote
 device>:<remote path>
scp share/renderdoc/plugins/aarch64/musl/lib/librenderdoc.so <remote
 device>:<remote path>
```

3. Setup the Vulkan layer files to the correct location on the server:

```
renderdoccmd vulkanlayer --register --<system or user>
```

4. Run the remote server.
5. Connect to the remote server in the RenderDoc **Remote Host Manager** dialog box:
   a. Go to **Tools > Manage Remote Servers**.
   b. In **Hostname**, enter the address of the remote server.
   c. Click **Add**.

   The server is then shown in the **Hostname** list. The icon next to your server shows the connection status of your server:

   - Connected
   - Not connected. Make sure that the server is accessible.

   If required, to refresh the list of servers, click **Refresh All**.
6. Close the **Remote Host Manager** dialog box.
   For more information, see the RenderDoc documentation at How do I capture and replay over a network?

Next steps

After connecting and configuring your Linux target, you can now perform an on-target capture using RenderDoc for Arm® GPUs.

- Capture frames from your application

2.3 Capture frames from your application

Set up and perform a capture on your target, ready for analysis using RenderDoc for Arm® GPUs.

Before you begin

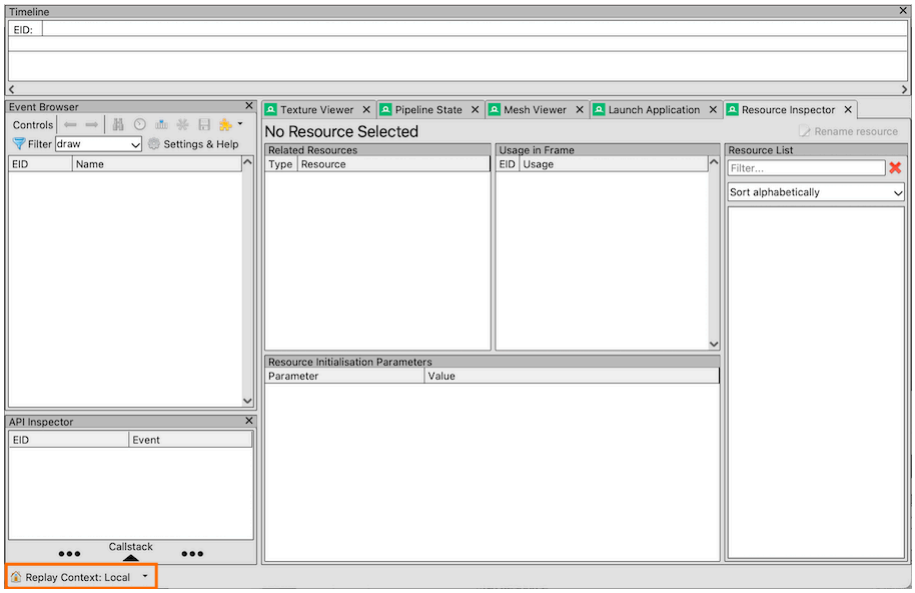
Before you begin this task, you must:

- Set up your target and system as described in Setup your Android target or Setup your Linux target.
- Ensure that your application is installed on your target.

Procedure

1. Connect your target to your host machine:
   - For Android only, ensure that adb can detect the target.
   - For Linux only, ensure that you can establish a TCP/IP connection to the target.
2. Select your connected target from the **Replay Context** dropdown list at the bottom left of the RenderDoc UI.

   Figure 2-1: Replay Context dropdown location in RenderDoc



   If you do not see your target listed in the dropdown list, check that you have set up the target correctly. See either Setup your Android target or Setup your Linux target.

Note

A red cross next to your target indicates that the target is disconnected.

   For Android only, when you connect for the first time, RenderDoc installs its capture and replay application on the target. Now the target is shown as connected in the dropdown list. After connecting, the RenderDoc APK starts running on your target.
3. In RenderDoc, navigate to the **Launch Application** tab, and set the following options:
   a. Set the **Executable Path** to the application that you want to debug. Click the **Browse** button to view all of the installed application packages on the target and find the `.exe` file:

- For Android only, choose the required application package folder and select the activity executable within it.
- For Linux only, select the executable file.

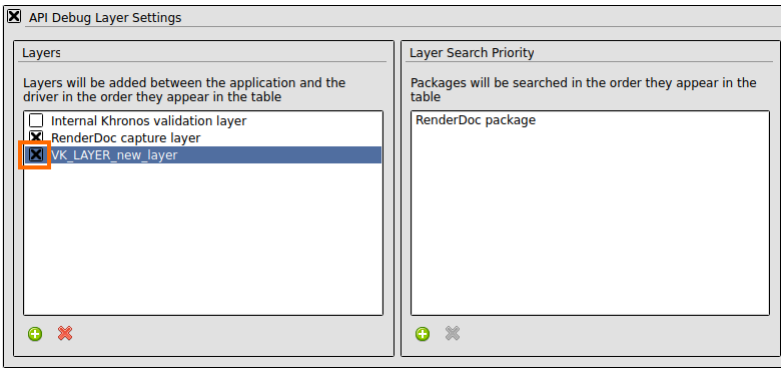
b. Optionally, you can specify the **Working Directory**. If you do not specify this, the capture is temporarily saved in the same location as the executable.

c. In the **Actions** section, you can also specify additional **Capture Options** and specify a list of frames to capture.

d. For Android 10.0 devices or later, RenderDoc for Arm® GPUs reads any Vulkan debug layers on the device and displays them in the **API Debug Layer Settings** section. Setup your Vulkan debug layers:

- Add or remove layers and packages for your capture session:
  - To add a new layer, click the **Add new layer** button located under the **Layers** table, then enter the layer name in the new row added to the table.
  - To add a new package, click the **Add new package** button located under the **Layer Search Priority** table, then select a package from the popup list.
  - To remove a layer or package, select the item that you want to remove, then click either the **Remove selected layer** button or **Remove selected package** button located under the appropriate table.
- To enable layers for your capture session, select the **Enabled** checkbox. Clear the **Enabled** checkbox for any layers that you do not want to use.

Figure 2-2: Enable layers

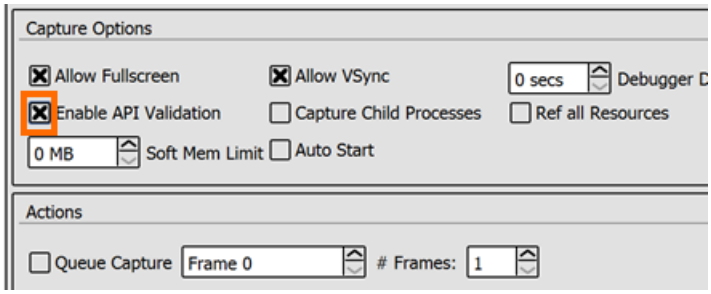


- The layers and packages are processed in the same order as they appear in the tables. Click and drag the layers and packages until they are in the required order.

Note

- You cannot edit or remove the included RenderDoc layer and RenderDoc package because they are required for the capture.
- To enable the validation layer, in the **Capture Options** dialog, select the **Enable API Validation** checkbox. Clear the **Enable API Validation** checkbox to disable this layer if you do not require it for your capture session.

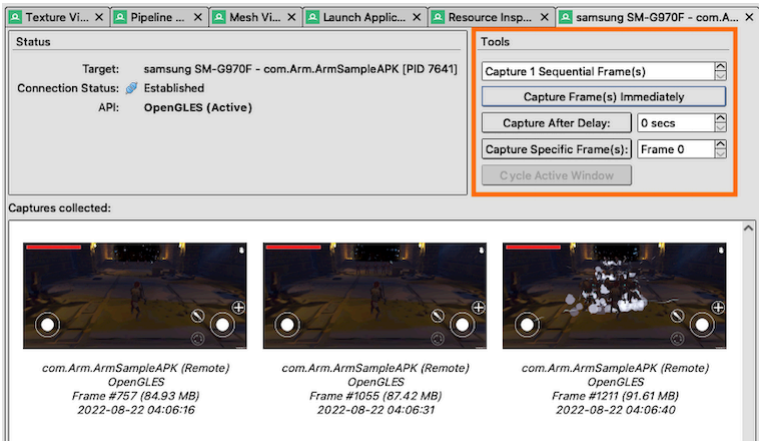
Figure 2-3: Enable API Validation



- Your capture will be unsuccessful if you add any new layers that do not exist.
- Layer settings persist between any captures that you open in the same capture session. When a device disconnects, or when RenderDoc closes, the layer settings on the device revert back to the same state they were in before the device connected to RenderDoc.

4. Click **Launch**, to start the application running on your target. After a successful launch, a new target-specific tab opens in the UI where you can select the frames that you want to capture:
   - Capture one or more frames immediately
   - Capture one or more frames after a delay
   - Capture one or more frames after a specific frame

Figure 2-4: Capture frame controls



Note

Captured frames are stored temporarily on the target.

5. When you have finished capturing the frames of interest, stop the application that you are debugging. Keep RenderDoc running though, so that you can analyze and debug your captures:

- For Android only, keep the RenderDoc APK running.
- For Linux only, keep the `renderdoccmd` running.

6. To open a capture, double-click on the thumbnail of the captured frame.

Next steps

When you have finished capturing, you can then analyze, debug, and edit your frames using RenderDoc.

- Analyze and debug your capture

2.4 Analyze and debug your capture

Use the debug features available in RenderDoc for Arm® GPUs to analyze and debug your Android capture.

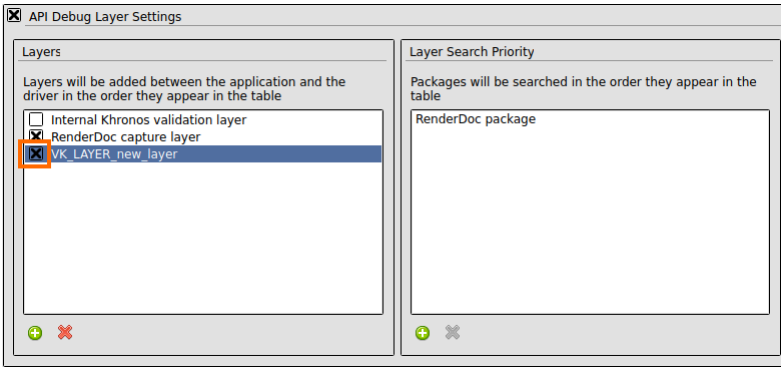
The primary purpose of RenderDoc is to help you diagnose rendering problems that occur in your application. When you have captured a frame, you can use the tool to interactively explore all of its API calls and rendering events. By stepping through the frame you can identify problem rendering events, and then review the configuration used by the event to discover the cause.

Load a saved capture

You can either load a frame capture for analysis directly after capture, or you can load a previously saved frame capture.

1. Ensure that your Android target is connected to your computer, and select your target from the **Replay Context** dropdown list.
2. If you have just taken a new frame capture, select the capture from the **Captures collected** window and click **Open**.
3. Alternatively, you can open a previous capture:
   - To load a previously saved frame capture, click **File > Open Capture**.
   - For Android 10.0 devices or later, you can setup Vulkan debug layers for your capture from the **File > Open Capture with Options** menu. The **API Debug Layer Settings** section shows the Vulkan layers and packages that RenderDoc for Arm® GPUs reads from your device:
     - Add or remove layers and packages for your capture:
       - To add a new layer, click the **Add new layer** button located under the **Layers** table, then enter the layer name in the new row added to the table.
       - To add a new package, click the **Add new package** button located under the **Layer Search Priority** table, then select a package from the popup list.
       - To remove a layer or package, select the item that you want to remove, then click either the **Remove selected layer** button or **Remove selected package** button located under the appropriate table.
     - To enable layers for your capture session, select the **Enabled** checkbox. Clear the **Enabled** checkbox for any layers that you do not want to use.

Figure 2-5: Enable layers

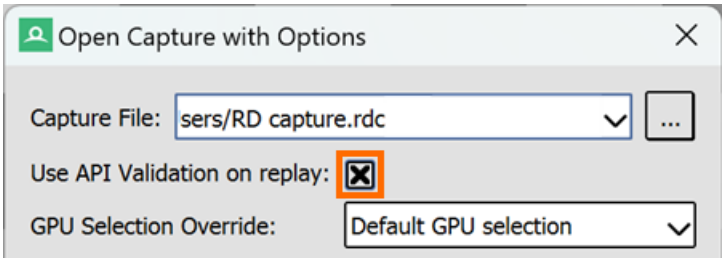


- The layers and packages are processed in the same order as they appear in the tables. Click and drag the layers and packages until they are in the required order.

Note

- To enable the validation layer, in the **Open Capture with Options** dialog, select the **Use API Validation on replay** checkbox. Clear the **Use API Validation on replay** checkbox to disable this layer if you do not require it.

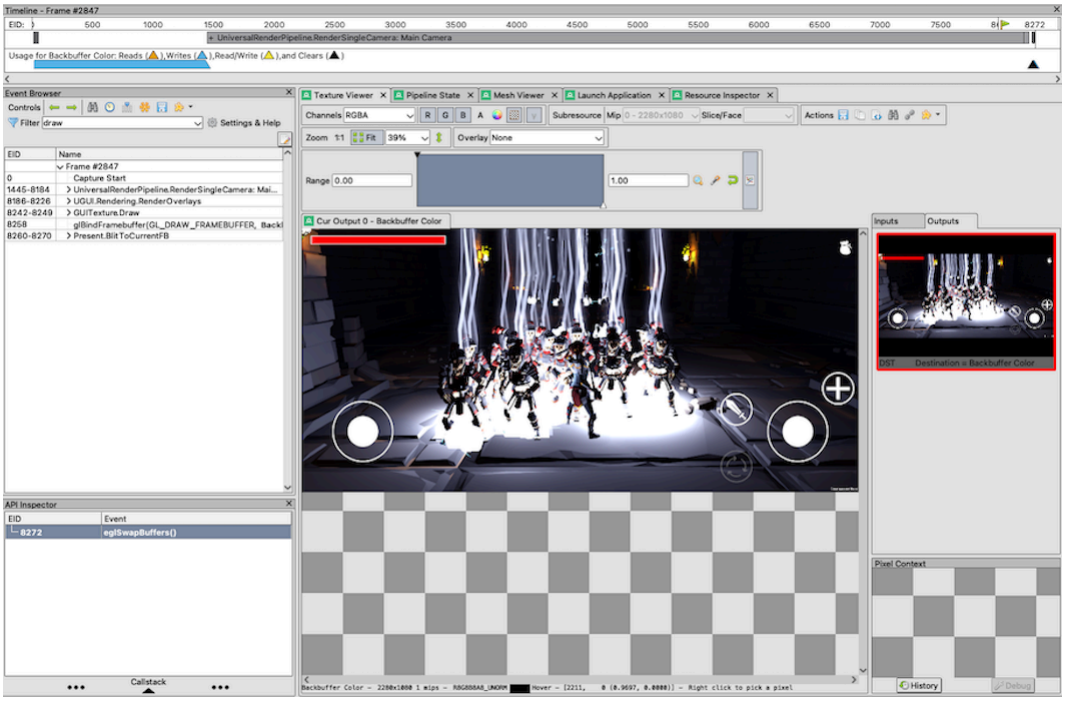
  Figure 2-6: Enable API Validation

- Your capture will be unsuccessful if you add any new layers that do not exist.
- Layer settings persist between any captures that you open in the same replay session. When a device disconnects, or when RenderDoc closes, the layer settings on the device revert back to the same state they were in before the device connected to RenderDoc.

When the frame has loaded, it is displayed on the target and in the **Texture Viewer** tab, and the **Event Browser** is populated in RenderDoc for Arm® GPUs.

Figure 2-7: RenderDoc UI with a loaded frame

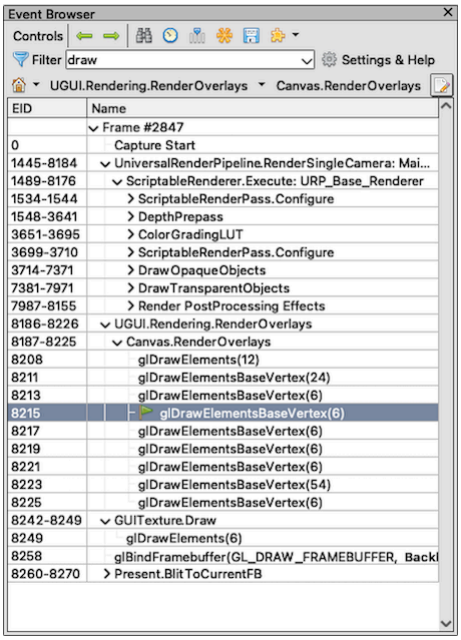


Navigate the frame capture

Use the **Event Browser** to navigate through the frame capture. By default, the **Event Browser** shows all `action()` events, which include draws, copies, and clears. Enter a search term in the **Filter** dropdown to filter these events.

Filter expressions can be complex. For more details, see the RenderDoc documentation at How do I filter visible events?.

Figure 2-8: Event browser view in RenderDoc



Selected events are highlighted with a green flag. All the other windows in the UI update to display information that is specific to the selected event. You can use this to view the render state and data resources that are used by the current event, and view the GPU output that resulted from it.

For more details, see the RenderDoc documentation at Event Browser

Debug a shader

Use the **Mesh Viewer** in RenderDoc to select an input vertex. Right-click anywhere in a row that is of interest, and select **Debug this Vertex** to open the vertex shader in the shader debugger.

Note

If the **Debug this Vertex** button is grayed out, this option might not be available for your target and API combination.

For more details, see the RenderDoc documentation at:

- How do I debug a shader?
- How do I use shader debug information?

Edit a shader

Shaders are one of the most important aspects of GPU processing, and errors in user shaders are a common cause of rendering problems. RenderDoc allows you to edit a shader used by an action event and replay it on the connected target. This feature allows you to quickly iterate changes without having to rebuild and deploy your entire application.

To launch the shader editor, click the **Pipeline State** tab then click the **Edit** button next to the shader. A text editor opens where you can make your edits to the code. To save and recompile the code, click **Apply changes**.

Figure 2-9: Edit shader button

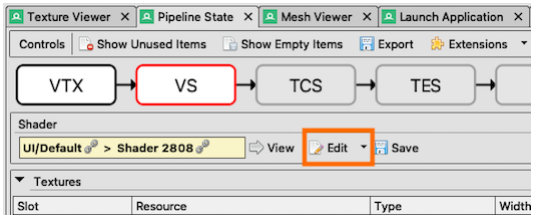


Figure 2-10: Editing the shader code

Note

Any changes to the shader will affect all action events that use this shader.

For more details, see the RenderDoc documentation at How do I edit a shader?

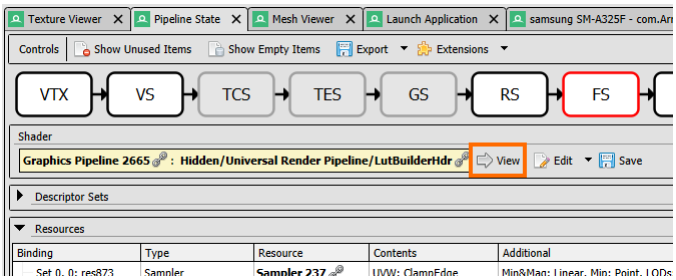
Review shader performance with Mali™ Offline Compiler

To review performance of your Vulkan shaders on the target GPU, use `malioc` to statically analyze the SPIR-V disassembly and generate a performance report from Mali™ Offline Compiler.

The report enables you to see details about the configuration, how the shader uses resources, performance cycle costs of the shader, and properties of the shader that can impact performance.

1. In the **Event Browser**, select a draw command.
2. Click the **Pipeline State** tab, select a shader, then click the **View** button.

Figure 2-11: View shader source



3. In the **Disassembly type** dropdown menu, and select the Mali Shader Performance report:
   - **Mali Shader Performance (malioc (Text Report))** opens a formatted and human readable text output of your shaders.
   - **Mali Shader Performance (malioc (JSON Report))** opens a machine-readable JSON format that you can export to other tools.

Figure 2-12: Disassembly_type_menu

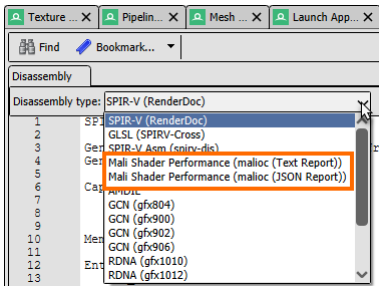
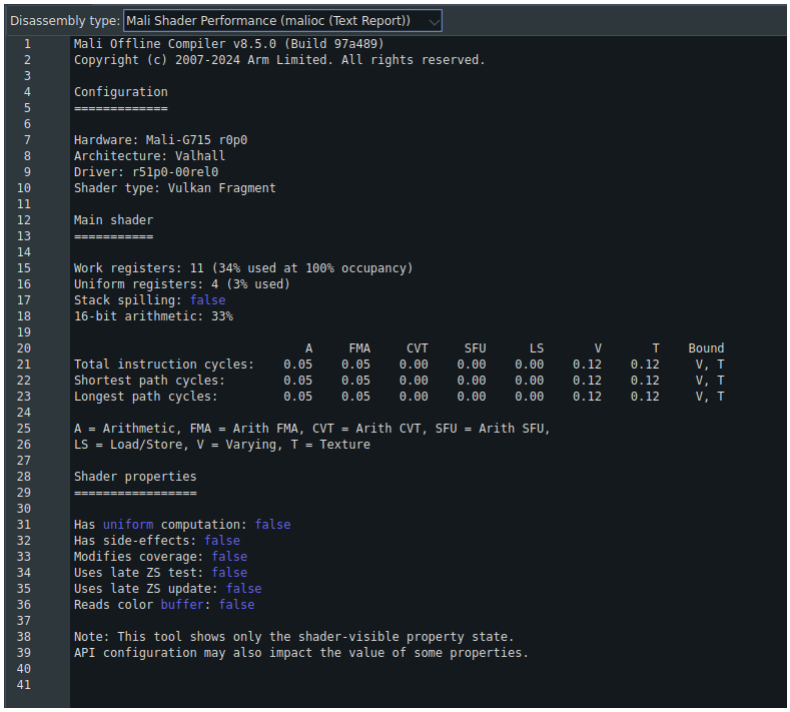


Figure 2-13: Example Mali Offline Compiler performance report



See the Arm® Mali™ Offline Compiler User Guide for more information about the metrics detailed in the Mali™ Offline Compiler performance report.

More things you can do with RenderDoc for Arm® GPUs

This section has described a few of the things you can do with RenderDoc for Arm® GPUs after you have captured a frame. See the RenderDoc documentation to explore the full list of features.

3. Troubleshooting RenderDoc

Find answers to problems that might occur when capturing or analyzing data in RenderDoc.

3.1 Capture fails to transfer over to the Android target

When you are connected to an Android target over USB, and you are loading a capture from a Linux or Mac host, the transfer of the capture to the target can fail on later versions of adb. You might also find that adb gets into a state where it cannot reconnect, and you must force it to stop.

A version of adb v34 or later disconnects the USB device mid-transfer

From adb version 34 or later, the default USB backend is `libusb`. You can check which USB backend that adb is using with the following command:

```
$ adb server-status
```

The following example shows an output returned for `libusb`:

```
usb_backend: LIBUSB
mdns_backend: OPENSCREEN
version: "35.0.2"
build: "12147458"
executable_absolute_path: "/tools/android-sdk/platform-tools/adb"
log_absolute_path: "/tmp/adb.28550.log"
os: "Linux 6.8.0-48-generic (x86_64)"
```

Solution

Start adb with the native USB backend enabled:

```
$ ADB_LIBUSB=0 adb start-server
```

The following example shows an output returned for the native USB backend:

```
usb_backend: NATIVE
usb_backend_forced: true
mdns_backend: OPENSCREEN
version: "35.0.2"
build: "12147458"
executable_absolute_path: "/tools/android-sdk/platform-tools/adb"
log_absolute_path: "/tmp/adb.28550.log"
os: "Linux 6.8.0-48-generic (x86_64)"
```

Proprietary Notice

This document is protected by copyright and other related rights and the use or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm Limited ("Arm"). No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, any patents, copyrights, trade secrets, trademarks, or other rights.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted use, duplication, or disclosure of this document complies fully with any relevant

export laws 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 this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

Product and document information

Read the information in these sections to understand the release status of the product and documentation, and the conventions used in Arm documents.

Product status

All products and services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

Product completeness status

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

Revision history

These sections can help you understand how the document has changed over time.

Document release information

The Document history table gives the issue number and the released date for each released issue of this document.

Document history

Issue	Date	Confidentiality	Change
2025.2-00	1 May 2025	Non-Confidential	New document for v2025.2
2025.1-00	20 March 2025	Non-Confidential	New document for v2025.1
2025.0-00	6 February 2025	Non-Confidential	New document for v2025.0
2024.6-00	28 November 2024	Non-Confidential	New document for v2024.6
2024.4-00	5 September 2024	Non-Confidential	New document for v2024.4

Change history

For information about the functional changes to RenderDoc for Arm® GPUs, see the *Arm® Performance Studio Release Notes*.

Conventions

The following subsections describe conventions used in Arm documents.

Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: developer.arm.com/glossary.

Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
italic	Citations.
bold	Interface elements, such as menu names. Terms in descriptive lists, where appropriate.
`monospace`	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace underline`	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`<and>`	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**.

We recommend the following. If you do not follow these recommendations your system might not work.

Your system requires the following. If you do not follow these requirements your system will not work.

You are at risk of causing permanent damage to your system or your equipment, or harming yourself.

This information is important and needs your attention.

A useful tip that might make it easier, better or faster to perform a task.

A reminder of something important that relates to the information you are reading.

Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Arm documents are available on developer.arm.com/documentation.

Confidential documents are only available to licensees, when logged in. Each document link in the tables below provides direct access to the online version of the document.

Arm product resources	Document ID	Confidentiality
Arm® Mali™ Offline Compiler User Guide	101863	Non-Confidential
Arm® Performance Studio Release Notes	107649	Non-Confidential
Download Arm Performance Studio for free	–	Non-Confidential

Non-Arm resources	Document ID	Organization
Android Debug Bridge	–	Android Developers
Android SDK platform tools	–	Android Developers
Android Studio	–	Android Developers
Developer Mode	–	Android Developers
Event Browser	–	RenderDoc
How do I capture and replay over a network?	–	RenderDoc
How do I debug a shader?	–	RenderDoc
How do I edit a shader?	–	RenderDoc
How do I filter visible events?	–	RenderDoc
How do I use RenderDoc on Android?	–	RenderDoc
How do I use shader debug information?	–	RenderDoc
How-to topics in the RenderDoc documentation	–	RenderDoc
RenderDoc Quick Start	–	RenderDoc
RenderDoc documentation	–	RenderDoc