Get started with Performance Advisor

1.1

Non-Confidential

Issue 00

102478_0101_00_en

Get started with Performance Advisor

Release information

Document history

Issue	Date	Confidentiality	Change
0101-00	27 June 2022	Non-Confidential	LWI directory update

Proprietary Notice

This document is protected by copyright and other related rights and the practice 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. 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 implementations infringe any third party patents.

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, has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

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.

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

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. Overview

To get quick analytics on how your Android application performs on your target device, use Performance Advisor to turn a Streamline capture into an easy-to-read performance summary.

This tutorial describes how to:

- Run the Python script `lwi_me.py` to set up the device
- Take a capture with Streamline
- Use the `pa` command to generate a Performance Advisor report in HTML format
- Use the `pa` command to generate a Performance Advisor report in JSON format (Professional edition only)
- Access command-line documentation for the `lwi_me.py` script and `pa` command.

The following video shows the steps required and gives an overview of a Performance Advisor report: Android profiling with Performance Advisor

Note

Before capturing with Streamline, you must use the provided Python script `lwi_me.py` to prepare your device. This script installs a daemon application and the relevant OpenGL ES or Vulkan layer library that enables frame data to be captured. You can not generate a Performance Advisor report from a Streamline capture without first running this script.

2. Before you begin

Do the following:

1. Download Arm Mobile Studio and follow the Installation instructions to install Arm Mobile Studio and set up your environment.
2. Ensure you have Android Debug Bridge (ADB) installed. 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. Performance Advisor uses a Python script to connect to your device. To run this script, you will need Python 3.5 or later installed.

3. Run the script and capture a Streamline profile

Performance Advisor runs on a Streamline capture file, so the first step is to take a capture with Streamline. In order to enable Streamline to capture the extra frame data from the device that Performance Advisor needs, you must run the provided Python script, `lwi_me.py`. This script does the following:

- Connects Streamline to your device and directs you to take a capture.
- Temporarily installs a daemon application on your device, called `gatord`, which Streamline uses to collect counter data.
- Temporarily installs the OpenGL ES or Vulkan layer library file on your device, which is needed to collect frame data.
- Enables you to specify options for the capture, such as whether to capture screenshots when the FPS drops below a certain threshold.

1. In a terminal window, navigate to the following folder in your Arm Mobile Studio installation directory, where the Python script `lwi_me.py` is located:

```
cd <installation_directory>/performance_advisor/bin/android
```

2. To run the script, simply type:

```
python3 lwi_me.py
```

Note

Do not move the script to another location, as it needs to be able to locate other files in your installation directory.

Optionally, you can use a range of different options to control the capture. For example, if you want to capture screenshots from the game when the FPS drops below 30, add the option `--lwi-mode` capture. You can specify a different threshold with `--lwi-fps-threshold=<val>`. You can also specify an output directory for the images with `--lwi-out-dir`. If you don’t specify a directory, Performance Advisor will create one in the current location.

```
python3 lwi_me.py --lwi-mode=capture --lwi-fps-threshold=50 --lwi-out-dir=$HOME/
Documents/MyGameProfile/SlowFrames
```

The script assumes that you are profiling a 64-bit OpenGL ES application. For 32-bit apps, you need to specify the `--32bit` option. For Vulkan apps, you need to add the `--lwi-api=vulkan` option.

For a full list of the available options, refer to the Performance Advisor user guide.

3. The script returns a numbered list of the Android package names for the debuggable applications that are installed on your device. Enter the number of the application you want to profile.

   The script identifies the GPU in the device, installs the daemon application, and waits for you to complete the capture in Streamline. Leave the terminal window open, as you will need to come back to it later to terminate the script.

4. Launch Streamline:

   a. On Windows, from the Start menu, navigate to Arm MS <version> and select Arm MS Streamline <version>.

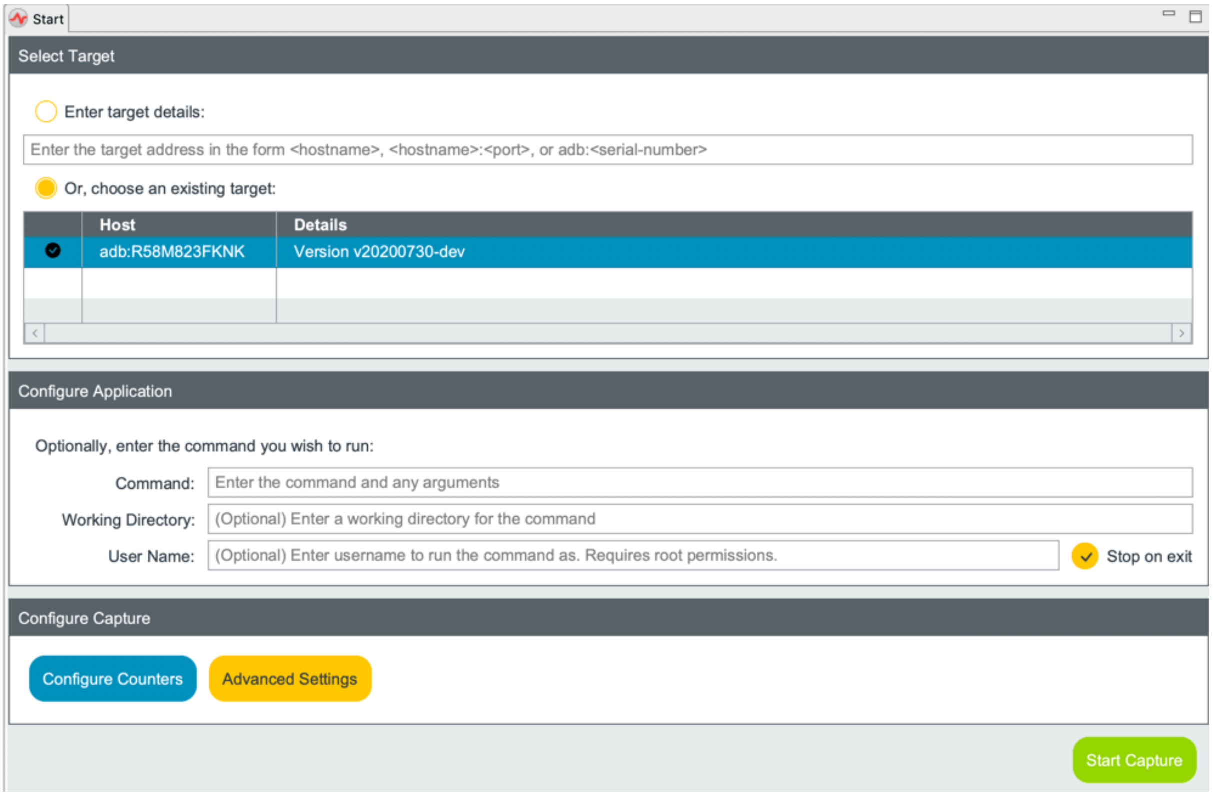
   b. On macOS, go to the `<install_directory>/streamline` folder, and double-click the `Streamline.app` file.

   c. On Linux, navigate to the `<install_directory>/streamline` directory in a terminal, and run the Streamline file:

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

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

Figure 3-1: Connect to your device in Streamline



○ Enter target details:

Enter the target address in the form <hostname>, <hostname>:<port>, or adb:<serial-number>

● Or, choose an existing target:

	Host	Details
✓	adb:R58M823FKNK	Version v20200730-dev

Optionally, enter the command you wish to run:

Command: Enter the command and any arguments

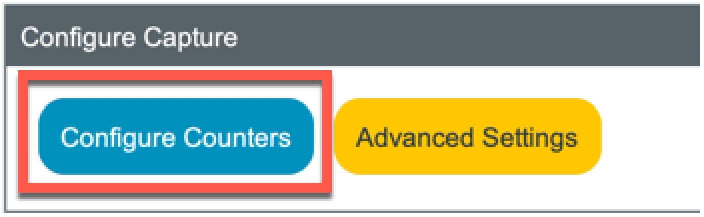
Working Directory: (Optional) Enter a working directory for the command

User Name: (Optional) Enter username to run the command as. Requires root permissions.

✓ Stop on exit

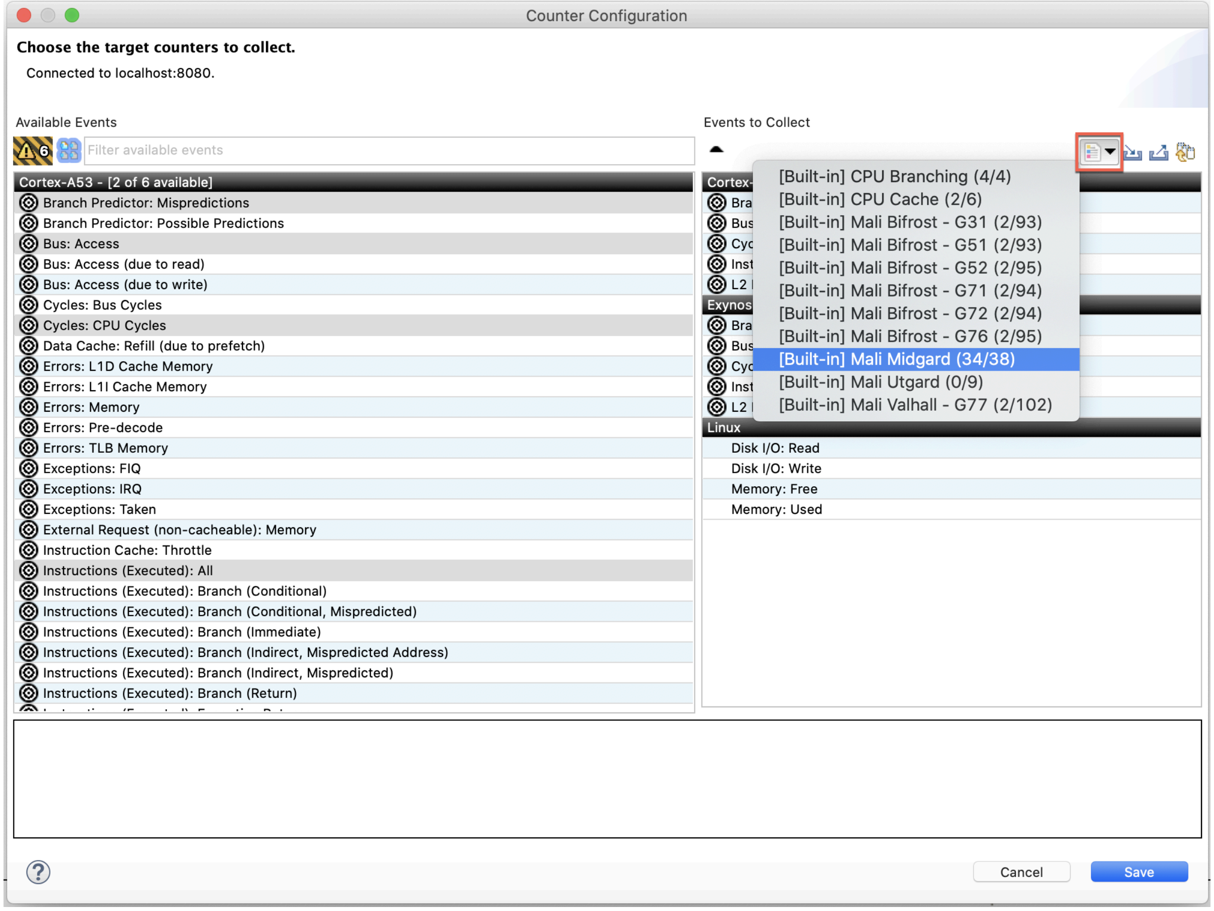
6. Open the Counter configuration dialog by selecting Configure Counters.

Figure 3-2: Streamline's configure counters button



7. Choose Add counters from a template and select an appropriate counter template for the GPU in your device.

Figure 3-3: Counter Configuration dialog



Choose the target counters to collect.

Connected to localhost:8080.

Available Events

Filter available events

Cortex-A53 - [2 of 6 available]
- Branch Predictor: Mispredictions
- Branch Predictor: Possible Predictions
- Bus: Access
- Bus: Access (due to read)
- Bus: Access (due to write)
- Cycles: Bus Cycles
- Cycles: CPU Cycles
- Data Cache: Refill (due to prefetch)
- Errors: L1D Cache Memory
- Errors: L1I Cache Memory
- Errors: Memory
- Errors: Pre-decode
- Errors: TLB Memory
- Exceptions: FIQ
- Exceptions: IRQ
- Exceptions: Taken
- External Request (non-cacheable): Memory
- Instruction Cache: Throttle
- Instructions (Executed): All
- Instructions (Executed): Branch (Conditional)
- Instructions (Executed): Branch (Conditional, Mispredicted)
- Instructions (Executed): Branch (Immediate)
- Instructions (Executed): Branch (Indirect, Mispredicted Address)
- Instructions (Executed): Branch (Indirect, Mispredicted)
- Instructions (Executed): Branch (Return)

Events to Collect

Cortex-
- Bra
- Bus
- Cyc
- Inst
- L2

Exynos
- Bra
- Bus
- Cyc
- Inst
- L2

Linux
- Disk I/O: Read
- Disk I/O: Write
- Memory: Free
- Memory: Used

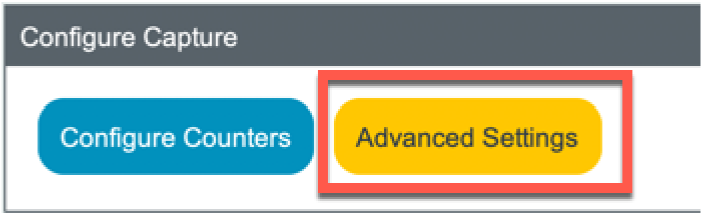
- [Built-in] CPU Branching (4/4)
- [Built-in] CPU Cache (2/6)
- [Built-in] Mali Bifrost - G31 (2/93)
- [Built-in] Mali Bifrost - G51 (2/93)
- [Built-in] Mali Bifrost - G52 (2/95)
- [Built-in] Mali Bifrost - G71 (2/94)
- [Built-in] Mali Bifrost - G72 (2/94)
- [Built-in] Mali Bifrost - G76 (2/95)
- [Built-in] Mali Midgard (34/38)
- [Built-in] Mali Utgard (0/9)
- [Built-in] Mali Valhall - G77 (2/102)

The number of counters in the template that your target device supports is shown next to each template. For example, here, 34 of the 38 available counters in the Mali Midgard template are supported in the connected device.

8. Save your selection.

9. Optionally, select Advanced Settings, to set additional capture options, including the sample rate and the capture duration.

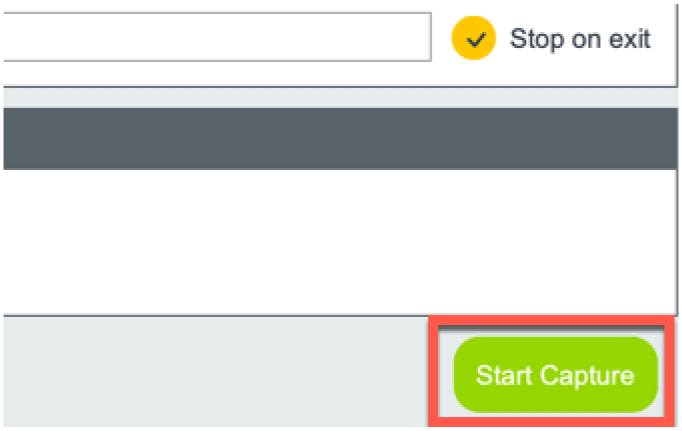
Figure 3-4: Streamline's advanced settings button



Refer to Capture options in the Arm Streamline User Guide for more details.

10. Select Start Capture.

Figure 3-5: Streamline start capture button



11. Specify the name and location of the capture file that Streamline will create when the capture is complete. Streamline then switches to Live view and waits for you to start the application on the device.
12. Start the application on the device and perform your test scenario.
13. Unless you specified a capture duration, click Stop capture to end the capture. Streamline stores the capture file in the location you specified.
14. IMPORTANT: Switch back to the terminal running the `lwi_me.py` script and press any key to terminate it. The script kills all processes that it started and remove the daemon application from the device.

Now that you have a Streamline capture, run Performance Advisor on it, to generate a performance report.

4. Generate a performance report

Do the following:

1. In a terminal, navigate to the location of your Streamline capture file. This file is usually a Streamline `.apc` directory but can also be a `.zip` file that has been exported from Streamline.

   ```
   cd $HOME/Documents/my_captures
   ```

2. Run Performance Advisor's `pa` command to generate an HTML report in the current location:

   ```
   pa my_capture.apc [options]
   ```

   You can pass options to the `pa` command, to control how it runs. For example, you can specify different output directory for the report, and metadata for the report, such as the application name, device name and build name, which are shown at the top of the report to help you identify the run:

   ```
   pa my_capture.apc --directory=$HOME/Documents/Reports --application-name="My
    Game" --device-name="A50" --build-name="First Build"
   ```

   If you captured screenshots of the game by specifying `--lwi-mode=capture` when running the script, you will need to supply the directory where those screenshots are saved, so that they will be shown on the report. Use the `--frame-capture` option to do this, for example, `--frame-capture=$HOME/Documents/MyGameProfile/SlowFrames`.

   For a full list of the available command options, refer to The pa command in the Performance Advisor user guide.

   Tip

   If you want to pass a number of command-line options to the `pa` command, you can create an options file containing them, which makes command-line entry much simpler.

   The HTML report is saved to the current location, or to the directory that you specified with the `--directory` option. Double-click this file to view it in a browser.

Note

If you are using an Arm Mobile Studio with a professional edition license to do system-wide profiling on a rooted device, you also need to specify the application name with the `--process` option:

```
pa my_capture.apc --process=app.package.name [options]
```

5. Generate a JSON file

Performance Advisor can generate report data in JSON format, so that you can import it into any JSON-compatible database, and visualize the data in your own performance dashboards, to track performance over multiple runs.

Note

This feature is available as part of Arm Mobile Studio professional edition. If you would like to integrate Performance Advisor into your continuous integration workflow, request a professional edition license.

1. Run Performance Advisor's `pa` command on your Streamline capture file, with the `--type` option to generate a JSON report called `report.json` in the current location:

   ```
   pa capture.apc --type=json:report.json
   ```

   You can specify both HTML and JSON types, to generate a report in each format:

   ```
   pa capture.apc --type=html:report.html,json:report.json
   ```

   Optionally, you can include options to specify a different directory to save the report to, and metadata such as the application name, device name and build name, to help you identify the run:

   ```
   pa capture.apc --directory=$HOME/Documents/Reports --application-name="My Game"
    --device-name="A50" --build-name="First Build"
   ```

   For a full list of the available command options, refer to The pa command in the Performance Advisor user guide.

   Note

   If you are using an Arm Mobile Studio to do system-wide profiling on a rooted device, you also need to specify the application name with the –process option:

   ```
   pa my_capture.apc --process=app.package.name [options]
   ```

2. The JSON report is saved to the current location, or to the directory that you specified with the `--directory` option.

For more information, refer to Integrate Arm Mobile Studio into a CI workflow.

6. Command-line options

The connection script `lwi_me.py` and the `pa` command both accept a range of command-line options to control how they run. Refer to the following topics in the Performance Advisor user guide for full descriptions of these options:

- lwi_me.py command-line options
- pa command-line options

Tip

If you want to pass a number of command-line options to the `pa` command, you can create an options file containing them, which makes command-line entry much simpler.

7. Related information

Here are some resources related to material in this guide:

- About Performance Advisor
- Optimization advice for mobile applications
- Setting performance budgets with Performance Advisor
- Integrate Arm Mobile Studio with Unreal Engine
- Integrate Arm Mobile Studio into a CI workflow
- Performance Advisor user guide