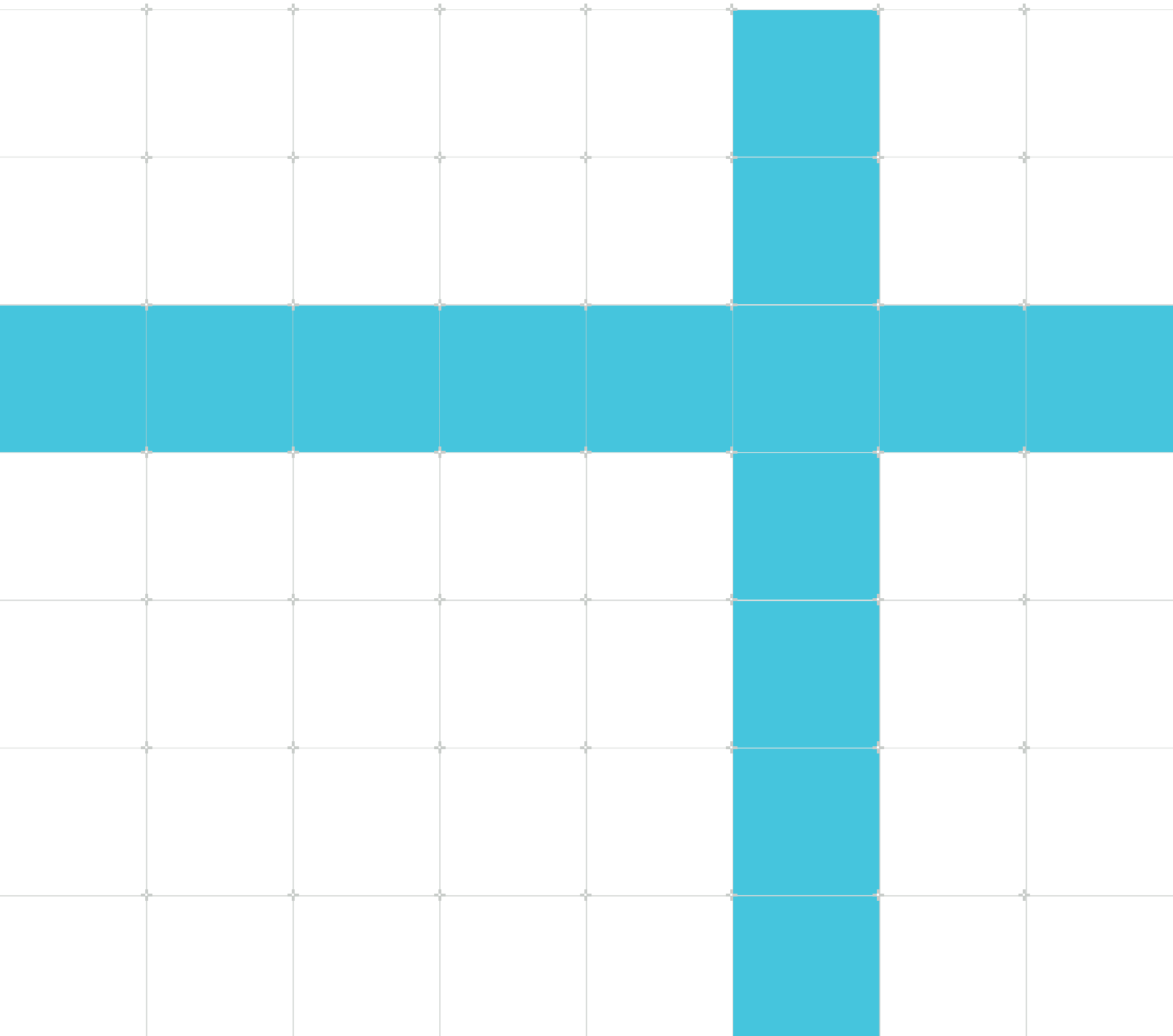
Arm® Streamline

Version 8.3

Target Setup Guide for Android

Non-Confidential
Copyright © 2021–2022 Arm Limited (or its affiliates).
All rights reserved.

Issue 00
101813_0803_00_en



Arm® Streamline

Target Setup Guide for Android

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.

Release information

Document history

Issue	Date	Confidentiality	Change
0708-00	20 August 2021	Non-Confidential	New document for v7.8
0709-00	18 November 2021	Non-Confidential	New document for v7.9
0800-00	18 February 2022	Non-Confidential	New document for v8.0
0801-00	20 May 2022	Non-Confidential	New document for v8.1
0802-00	19 August 2022	Non-Confidential	New document for v8.2
0803-00	17 November 2022	Non-Confidential	New document for v8.3

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, and has undertaken no analysis to identify or understand the scope and content of, 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 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.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm's trademark usage guidelines at https://www.arm.com/company/policies/trademarks.

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

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.

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Contents

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

1. Introduction

Learn how to use Streamline to set up your Android device for debugging and profiling.

1.1 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**.
Caution	Recommendations. Not following these recommendations might lead to system failure or damage.
Warning	Requirements for the system. Not following these requirements might result in system failure or damage.
Danger	Requirements for the system. Not following these requirements will result in system failure or damage.
Note	An important piece of information that needs your attention.

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Convention	Use
	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.

1.2 Useful resources

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

Access to Arm documents depends on their confidentiality:

- Non-Confidential documents are available at developer.arm.com/documentation. Each document link in the following tables goes to the online version of the document.
- Confidential documents are available to licensees only through the product package.

Arm product resources	Document ID	Confidentiality
Arm Development Studio Getting Started Guide	101469	Non-Confidential
Arm Streamline User Guide	101816	Non-Confidential

Non-Arm resources	Document ID	Organization
Debug your app	-	Android Studio

1.3 Other information

See the Arm website for other relevant information.

- Arm® Developer.
- Arm® Documentation.
- Technical Support.
- Arm® Glossary.

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

2. Target Setup

This chapter explains how to get your application, device and host machine ready to use Streamline for application or system profiling.

2.1 Application and system profiling

Streamline supports two types of profiling. Application profiling is the most common use case, but system profiling is also supported.

Application profiling

Streamline supports data capture on a non-rooted Android device. Streamline collects CPU performance data and Arm® Mali™ GPU, or Arm Immortalis™ GPU, performance data from a single application, so you can profile your game or application without device modification. To configure Streamline to collect the right data, use the templates to select the most appropriate set of counters for your device.

System profiling

In addition to the single application profiling for non-root devices, Streamline also supports system-wide Android profiling when running on development devices with root access. System profiling enables manufacturers to simultaneously monitor all applications and services running on their device, allowing identification of problematic processes or scheduling behaviors.

Related information

Debuggable application profiling on page 13
System profiling on page 18

2.2 Compile your application

Before you can profile your executable with Streamline, you must compile your executable. This topic describes the compiler options to use when you compile your application.

Profiling native code

When compiling with GCC or Clang, use the following options:

`-g`
: Turns on the debug symbols necessary for quality analysis reports.

`-fno-inline`
: Disables inlining and substantially improves the call path quality.

`-fno-omit-frame-pointer`
: Compiles your EABI images and libraries with frame pointers. This option enables Streamline to record the call stack with each sample taken.

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

`-mno-omit-leaf-frame-pointer`

Keeps the frame pointer in leaf functions.

`-marm`

When building for AArch32, if GCC was compiled with the `--with-mode=thumb` option enabled, this `-marm` is required. Using `--with-mode=thumb` without `-marm` breaks call stack unwinding in Streamline.

Optional compiler options for call stack unwinding

To enable call stack unwinding in Streamline, you need to compile your executable with some additional compiler options:

- For AArch64 applications:
  - Compiling with GCC, use: `-fno-omit-frame-pointer` and `-mno-omit-leaf-frame-pointer`

    
    Note

    Arm recommends using `-mno-omit-leaf-frame-pointer` to prevent samples in leaf functions incorrectly listing their grand-parent function as their parent.

  - Compiling with Clang, use: `-fno-omit-frame-pointer`

    
    Note

    `-mno-omit-leaf-frame-pointer` is not supported on Clang.

- For AArch32 applications, compiling with either GCC or Clang, use: `-fno-omit-frame-pointer`, `-marm`, and `-mapcs-frame`.

  
  Note

  Streamline supports call stack unwinding for code that has been generated by Arm® Compiler 6.

  Streamline does not support call stack unwinding for T32 (Thumb®) code.

Profiling Java or Kotlin code

In addition to profiling native code, Streamline can profile OAT files that Android runtime (ART) generates. The Streamline report for the application shows function names and disassembly in the **Code** view, but not source code.

To enable OAT files to be built with debug symbols, you must run `dex2oat` with the `--no-strip-symbols` option. To run `dex2oat` with the `--no-strip-symbols` option:

1. Run the following command on the device:

```
setprop dalvik.vm.dex2oat-flags --no-strip-symbols
```

2. Re-install the APK file

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

3. To verify the options for `dex2oat` are set correctly, run the command:

```
getprop dalvik.vm.dex2oat-flags
```

4. To check whether DEX files contain `.debug_*` sections, you can use the GNU tools `readelf` command, for example:

```
readelf -S .../images/*.dex
```

Source code annotations

To enable Streamline to provide extra context when profiling your executable, you can add annotations to your source code. Streamline supports two types of annotations:

- User space annotations, for annotating your application
- Kernel annotations, to profile system calls

You can read more information about annotating your code in the Annotate your code chapter of the *Arm Streamline User Guide*.

2.3 Set up your host machine

Arm® Streamline Performance Analyzer is available for the Arm Mobile Studio or the Arm Development Studio product suites. To use Streamline, install the necessary software and set up environment variables on your host machine.

Before you begin

- Ensure you have a device that is correctly configured to generate performance data. You can use many consumer Android devices without modification. A list of the consumer devices that Arm have tested to support Streamline is available from Supported Devices.
- If you are building your own device software, ensure that your kernel configuration includes the options that are described in Kernel configuration menu options .

1. Ensure you have installed Python 3.6 (or higher). Arm Mobile Studio uses Python to run the provided `lwi_me.py` and `gator_me.py` scripts, which uses the `gatord` agent to connect Streamline to your Android target.
2. Ensure you have installed Android Debug Bridge (ADB). ADB is available with the Android SDK platform tools, which are installed as part of Android Studio. Alternatively, you can download the latest version of ADB from the Android SDK platform tools site.

Procedure

1. Download the studio package appropriate to your host platform (Windows, Linux, or macOS):
   - Download Arm Mobile Studio from Downloads
   - Download Arm Development Studio from Downloads
2. Install your studio package:

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

- To install Arm Mobile Studio, use the instructions in the Arm Streamline User Guide
- To install Arm Development Studio, use the instructions in the Arm Development Studio Getting Started Guide

3. Add the path to the Android SDK and Python executables to your `PATH` environment variable.

Next steps

- Set up your device
- Profile your application

2.4 Set up your device

To use Streamline, set up an Android device with the application you want to profile.

Before you begin

Set up your host machine

Procedure

1. Ensure that Developer Mode is enabled, then enable USB Debugging by selecting **Settings > Developer options**.
2. Connect the device to the host through USB and approve the debug connection when prompted. If the connection is successful, running the `adb devices` command on the host returns the ID of your device, and you can run `adb shell`.
3. Build and install the debuggable application for profiling:
   - For Unity applications, select the **Development Build** option in the **Build Settings**.
   - For applications that are not built with Unity, ensure it is marked as debuggable in the Android application manifest. See how to debug your application in the Android Studio documentation.

Next steps

Profile your application

2.5 Launch Streamline

Learn how to open Streamline using different operating systems.

Before you begin

Before you can open and use Streamline, ensure you have followed the steps in Set up your host machine and Set up your device.

Procedure

Launch Streamline:

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

- On Windows:
  - Arm® Mobile Studio users: From the **Start** menu, search for **Streamline** and select the **Streamline** shortcut.
  - Arm Development Studio users: Launch Arm Development Studio, navigate to the **Streamline data** view, and click **Launch Streamline**.

If required, to open the **Streamline data** view, either:

- Search for 'Streamline data' in the Arm Development Studio search function, and then select the **Streamline data** view.
- Select **Window > Show view > Other...**, expand **Streamline**, select **Streamline Data**, and click **Open**.

- On macOS, go to the `<install_directory>/streamline` folder, and double-click the `Streamline.app` file.
- On Linux:
  - Arm Mobile Studio users: Navigate to the `<install_directory>/streamline` folder and run the `Streamline` file:

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

  - Arm Development Studio users: Launch Arm Development Studio, navigate to the **Streamline data** view, and click **Launch Streamline**.

If required, to open the **Streamline data** view, either:

- Search for 'Streamline data' in the Arm Development Studio search function, and then select the **Streamline data** view.
- Select **Window > Show view > Other...**, expand **Streamline**, select **Streamline Data**, and click **Open**.

Next steps

- Connect Streamline to your device

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

3. Debuggable application profiling

Profile your application while it is running on a non-rooted Android device.

3.1 Profile your application

Capture a profile of a debuggable application running on an unrooted Android device.

3.1.1 Connect Streamline to your device

Use the **Start** view to connect Streamline to an Android device and collect data from a debuggable application.

Before you begin

Before you can connect Streamline to a device, ensure you have followed the steps in Set up your device.

Procedure

1. Launch Streamline.
2. In the **Start** view, select **Android (adb)** as your device type, then select your device from the list of detected devices. If you do not see your device in the list, check that it is connected properly through USB. See Set up your device for more information.
3. Wait a few moments for the list of available packages to populate, then select the package you want to profile from the list of packages available on the selected device.

Next steps

Choose a counter template. For more information about how to find and select a counter template, see Choose a counter template .

3.1.2 Choose a counter template

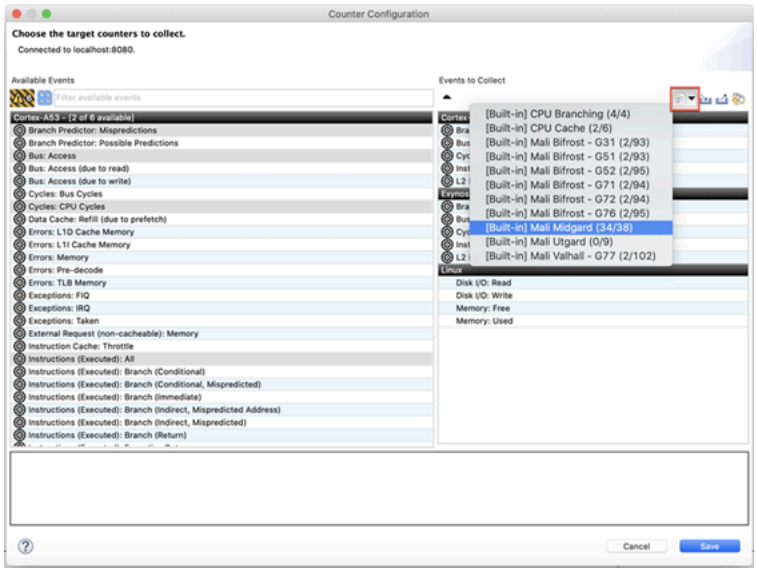
Counter templates are pre-defined sets of counters that enable you to review the performance of both CPU and GPU behavior. Choose the most appropriate template for the GPU in your device.

Procedure

1. In the **Start** view, click **Configure Counters**.
2. Click **Add counters from a template** to see a list of available templates.

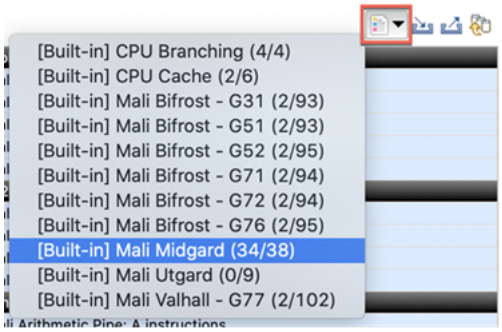
Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-1: Templates available from the Counter Configuration dialog box.



3. Select a counter template appropriate for the GPU in your device, then **Save** your changes. The number of counters in the template that your device supports is shown next to each template. For example, here, 34 of the 38 available counters in the Arm® Mali™ Midgard template are supported in the connected device.

Figure 3-2: Choose a counter template appropriate to the GPU in your device.



4. Optionally, in the **Start** view, click **Advanced Settings** to set more capture options, including the sample rate and the capture duration (by default unlimited). Refer to Set capture options in the *Arm Streamline User Guide*.

Next steps

Capture a profile using Streamline. For more information about how to capture the behavior of your CPU and GPU performance using Streamline, see Capture a profile.

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

3.1.3 Capture a profile

Start a capture session to profile data from your application in real time. When the capture session ends, Streamline automatically opens a report for you to analyze later.

Before you begin

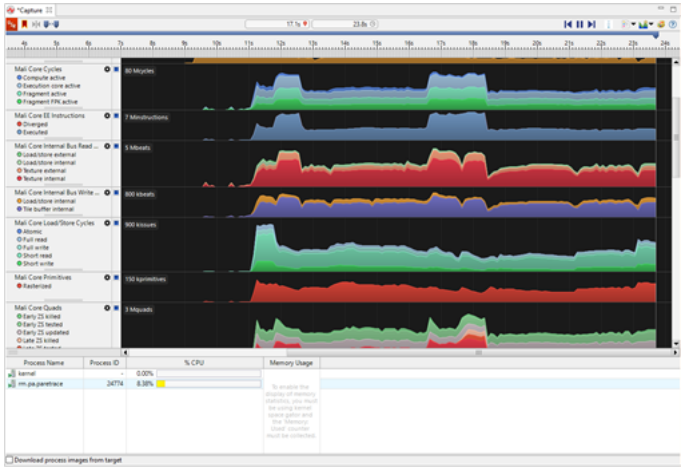
Before you capture a profile, ensure you have followed the steps in:

- Set up your device
- Connect Streamline to your device

Procedure

1. In the **Start** view, click **Start Capture** to start capturing data from your device.
   Specify the name and location on the host of the capture file that Streamline will create when the capture is complete. Streamline then switches to the **Live** view and waits for you to start the application on the device.
2. Start the application that you want to profile.
   The **Live** view shows charts for each counter that you selected. Below the charts is a list of running processes in your application with their CPU usage. The charts now start updating in real time to show the data that Streamline captures from your running application.

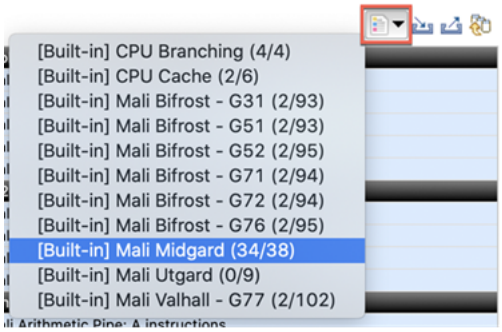
   Figure 3-3: Live view shows charts capturing data from your running application.



3. Unless you specified a capture duration, in the **Capture Control** view, click **Stop capture and analyze** to end the capture.
   Streamline stores the capture file in the location that you specified previously, and then prepares the capture for analysis. When complete, the capture appears in the **Timeline** view.
4. Click **Switch and manage templates** and select the same counter configuration template that you chose to create the capture.

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-4: Choose a counter template appropriate to the target GPU in your device.



Next steps

Analyze the data. For more information about how to analyze performance with Streamline, see Analyze your capture in the *Arm Streamline User Guide*.

3.2 Generate a headless capture

When integrating performance analysis into continuous integration, capturing data without having the host tool connected or a user manually controlling the GUI is often required. To capture data without the Streamline host tool connected, use the `gator_me.py` script in headless mode.

Before you begin

- Install Python 3.5 (or higher). You need Python 3.5 or higher to run the provided `gator_me.py` script, which uses the `gatord` agent to connect Streamline to your Android device.
- Add the path to the `Python3` directory to your `PATH` environment variable.
- Configure counters and export a counter configuration file for each class of device

Procedure

1. On the host, run the `gator_me.py` Python script to set up the device for a headless data capture.

   ```
   python3 gator_me.py --package <your_app_package> --daemon <path_to_gatord> --
   config <path_to_your_configuration.xml> --headless <output.apc.zip>
   ```

   The script is in the following directory:

   ```
   <install_directory>/streamline/gator/
   ```

   Use the following command-line arguments:

   - The Android package name of the application that you want to profile.
   - The path on the host to the `gatord` binary to install on the device. By default, this path is the current working directory. Your installation provides two versions of `gatord`, in the following directories:

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

    - `<install_directory>/streamline/bin/android/arm/` for 32-bit architectures.
    - `<install_directory>/streamline/bin/android/arm64/` for 64-bit architectures.
  - The path to the configuration file that you saved in the **Before you begin** steps.
  - The path to store the saved output file to.
2. Run your test scenario and exit the application when it has completed.
3. Wait for the script to download the data from the device, and write out the `output.apc.zip` file.
   The script stops automatically when it detects that the application is no longer running.
4. To view the data in the Streamline GUI, start the host application and import the APC file into the **Streamline Data** view.

Next steps

Analyze the data. For more information about how to analyze performance with Streamline, see Analyze your capture in the *Arm Streamline User Guide*.

Related information

Capture a Streamline profile

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

4. System profiling

Profile all applications and services that are running on a rooted Android device.

4.1 Profile your system

Set up and run Streamline with Android device root user access.

Before you begin

- Set up your host machine
- Set up your device

Procedure

1. In a command terminal, run `gatord` as root:

```
adb push gatord /data/local/tmp
adb shell
cd /data/local/tmp
su
./gatord --system-wide=yes
```

2. Launch Streamline.
3. Open the **Start** view, and select **TCP (Advanced)** as your device type.
4. Select your device by entering the address or by using `adb <serial-number>`. Alternatively, select your device from the list of detected devices.

Next steps

Choose a counter template. For more information about how to find and select a counter template, see Choose a counter template.

4.2 Enable atrace annotations

Streamline can capture Android trace points that atrace generates.

Before you begin

- To collect atrace, your Android device must be rooted and use a Linux kernel version of 3.10 or later.
- To notify running applications that atrace annotation tags are enabled, you must install the `notify.dex` file on your device in the same directory as `gatord`. The Java source code for `notify.dex` is available from either:
  - `<install_directory>/streamline/gator/notify/`
  - The `notify` directory in https://github.com/ARM-software/gator

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Procedure

1. In the **Start** view, select your device type as **TCP (advanced)**.

TCP (advanced) device type is required to collect and configure atrace counters.

2. Select your device and click **Configure counters**.
3. Drag the events from the **Atrace** counter section from the **Available Events** column to the **Events to Collect** column.

If you expect to see atrace events in this dialog box but none are displayed, click the **Warnings** tag to see why atrace support is disabled. Ensure your device is compatible and configured as described in the **Before you begin** section of this topic.

4. Click **Save**.

Next steps

Start the capture. In the Streamline GUI, click **Start capture**.

In your capture, string annotations or counter charts are created from your atrace macros. You can learn more about annotations and where they appear in the **Timeline** and **Log** views in the Annotate your code section of the the Streamline User Guide.

Related information

Annotate your code

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

5. Advanced device setup information

This appendix provides extra configuration information beyond the standard setup.

5.1 Kernel configuration menu options

You must enable certain kernel configuration options to run Streamline.

The following `menuconfig` menus have options that are required for Streamline:

- If these options are not set correctly, you must change them and rebuild your kernel. If they are set correctly, you are ready to build and install the gator driver.
- The location of these options might change between releases. If so, use the search option in `menuconfig` to find them.
- Extra options are required to enable Arm® Mali™ GPU support.

General Setup

Enable the **Profiling Support** option `CONFIG_PROFILING`, and the **Kernel performance events and counters** option `CONFIG_PERF_EVENTS`. `CONFIG_PERF_EVENTS` is required for kernel versions 3.0 and later. Enable the **Timers subsystem** > **High Resolution Timer Support** option `CONFIG_HIGH_RES_TIMERS` .

Kernel Features

The **Enable hardware performance counter support for perf events** option `CONFIG_HW_PERF_EVENTS`. `CONFIG_HW_PERF_EVENTS` is required for kernel versions 3.0 and later. If you are using Symmetric MultiProcessing (SMP), enable the **Use local timer interrupts** option `CONFIG_LOCAL_TIMERS`. If you are running on Linux version 3.12 or later, the `CONFIG_LOCAL_TIMERS` option is not necessary.

CPU Power Management

Optionally enable the **CPU Frequency scaling** option `CONFIG_CPU_FREQ` to enable the CPU Freq chart in the **Timeline** view. `gatord` requires kernel version 2.6.38 or greater to enable this chart.

Kernel hacking

If other trace configuration options are enabled, the **Trace process context switches and events** option `CONFIG_ENABLE_DEFAULT_TRACERS` might not be visible in `menuconfig` as an option. Enabling one of these other trace configurations, for example `CONFIG_GENERIC_TRACER`, `CONFIG_TRACING`, or `CONFIG_CONTEXT_SWITCH_TRACER`, is sufficient to enable tracing. Optionally enable the **Compile the kernel with debug info** option `CONFIG_DEBUG_INFO`. This option is only required for profiling the Linux kernel.

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Kernel versions before 4.6, with `CONFIG_CPU_PM` enabled, produce invalid results. For example, counters not showing any data, large spikes, and non-sensible values for counters. This issue is due to the kernel PMU driver not saving state when the processor powers down, or not restoring state when it powers up. To avoid this issue, upgrade to the latest version of the kernel, or apply the patch found at https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=da4e4f18afe0f3729d68f3785c5802f786d36e34. This patch applies cleanly to version 4.4, and it might also be possible to back port it to other versions. If you apply the patch, you might also require the patch at https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=cbcc72e037b8a3eb1fad3c1ae22021df21c97a51.

5.2 Build gatord yourself

Build `gatord` yourself to apply patches for bug fixes or add support for new features.

About this task

It is not possible to build `gatord` on a Windows host.

Procedure

1. Either download the `gatord` source from the daemon directory in https://github.com/ARM-software/gator, or copy the source that is supplied in `<DS_install_directory>/sw/streamline/gator/daemon/`.
2. Follow the instructions in the `README.md` file in the `gator` directory.

5.3 gatord command-line options

`gatord` must be running before you can capture trace data. The command-line options configure how `gatord` captures events and how it communicates with Streamline running on your host.

`gatord` has two modes of operation:

Daemon mode (the default mode)

Sends captured events to a host running Streamline.

Local capture mode

Writes the capture to a file then exits.

To enable this mode, specify an output directory with the `--output` flag.

Arguments available to all modes:

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table 5-1: gatord arguments available to all modes

Option	Description
`-h, --help`	Lists all the available `gatord` command-line options.
`-c, --config-xml <config_xml>`	Specify the path and filename of the `configuration.xml` file that defines the capture options. In daemon mode, the list of counters is written to this file. In local capture mode, the list of counters is read from this file.
`-e, --events-xml <events_xml>`	Specify the path and filename of the `events.xml` file. `events.xml` defines all the counters that Streamline collects during the capture session.
`-E, --append-events-xml <events_xml>`	Specify the path and filename of the `events.xml` file to append.
`-P, --pmus-xml <pmu_xml>`	Specify the path and filename of the `pmu.xml` file to append.
`-v, --version`	Print version information.
`-d, --debug`	Enable debug messages.
`-A, --app <cmd> <args...>`	Specify the command to execute when the capture starts. This argument must be the last argument that is passed to `gatord`. All subsequent arguments are passed to the launched application.
`-k, --exclude-kernel <yes|no>`	Specify whether to filter out kernel events from the perf results.
`-S, --system-wide <yes|no>`	Specify whether to capture the whole system. In daemon mode, `no` is only applicable when `--allow-command` is specified. In this mode, you must enter a command in the **Start** view. Requires kernel events to be enabled in **Capture and Analysis Options of Streamline**, or by setting `--exclude-kernel no`. Defaults to `yes`, unless `--app`, `--pid`, or `--wait-process` is specified.
`-u, --call-stack-unwinding <yes|no>`	Enable or disable call stack unwinding. Defaults to `yes`.
`-r, --sample-rate <none|low|normal|high>`	Specify the sample rate for the capture. The frequencies for each sample rate are: • `high` = 10kHz • `normal` = 1kHz • `low` = 100Hz • `none` = the lowest possible rate Defaults to `normal`.
`-t, --max-duration <s>`	Specify the maximum duration that the capture can run for in seconds. Defaults to `0`, which means unlimited.
`-f, --use-efficient-ftrace <yes|no>`	Enable efficient ftrace data collection mode. Defaults to `yes`.
`-w, --app-cwd <path>`	Specify the working directory for the application that `gatord` launches. Defaults to the current directory.
`-x, --stop-on-exit <yes|no>`	Stop the capture when the launched application exits. Defaults to `no`, unless `--app`, `--pid`, or `--wait-process` is specified.
`-Q, --wait-process <command>`	Wait for a process that matches the specified command to launch before starting the capture. Attach to the specified process and profile it.
`-Z, --mmap-pages <n>`	The maximum number of pages to map per mmaped perf buffer is equal to `<n+1>`. `n` must be a power of two.

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Option	Description
`-O, --disable-cpu-onlining <yes|no>`	To not switch on CPU cores that are offline to read their information. This option is useful for kernels that fail to handle this action correctly, for example they reboot the system. Defaults to `no`.
`-F, --spe-sample-rate <n>`	Specify the SPE periodic sampling rate. The rate, `<n>`, is the number of operations between each sample, and must be a nonzero positive integer. The hardware specifies the minimum rate. Values below this threshold are ignored and the hardware minimum is used instead.
`--smmuv3-model <model_id>|<iidr>`	Specify the SMMUv3 model. You can specify the model ID string directly, such as `mmu-600`, or the hex value representation for the model's IIDR number either fully, such as `4832243b`, or partially, such as `483_43b`.

Arguments available on Android devices only:

Table 5-2: gatord arguments available on Android devices only

Option	Description
• `l, --android-pkg <pkg>`	Profiles the specified android package. Waits for the package app to launch before starting a capture unless `--android-activity` is specified.
• `m, --android-activity <activity>`	Launch the specified activity of a package and profile its process. You must also specify `--android-pkg`.

Arguments available in daemon mode only:

Table 5-3: gatord arguments available in daemon mode only

Option	Description
`-p, --port <port_number>|uds`	Set the port number that `gatord` uses to communicate with the host. The default is 8080. If you use the argument `uds`, the TCP socket is disabled and an abstract Unix domain socket is created. This socket is named `streamline-data`. If you use Android, creating a Unix domain socket is useful because `gatord` is usually prevented from creating a TCP server socket. Alternatively, you can connect to `localhost:<local_port>` in Streamline using: `adb forward tcp:<local_port> localabstract:streamline-data`
`-a, --allow-command`	Allows you to run a command on the device during profiling. The command is specified in the **Start** view. **Caution:** If you use this option, an unauthenticated user could run arbitrary commands on the device using Streamline.

Arguments available to local capture mode only:

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Table 5-4: gatord arguments available to local capture mode only

Option	Description
`-s, --session-xml <session_xml>`	Specify the `session.xml` file that the configuration is taken from. Any additional arguments override values that are specified in this file.
`-o, --output <apc_dir>`	Specify the path and filename of the output directory for a local capture. The directory path will be appended with the extension `.apc` if it is not already the case.
`-i, --pid <pids...>`	A comma-separated list of process IDs to profile.
`-C, --counters <counters>`	A comma-separated list of counters to enable. You can specify this option multiple times. An event code and a slot identify most hardware counters. To specify the counter for a particular slot, pass: `--counters <device>_cnt<s>:<e>` Where: • `<device>` is the prefix that identifies the device type. • `<s>` is the slot number. • `<e>` is the event code.
`-X, --spe <id>[:events=<indexes>][:ops=<types>][:min_latency=<lat>]`	Enable the Statistical Profiling Extension (SPE). Where: • `<id>` is the name of the SPE properties that are specified in the `events.xml` or `pmus.xml` file. It uniquely identifies the available events and counters for the SPE hardware. • `<indexes>` is a comma-separated list of event indexes to filter the sampling by. A sample is only recorded if all events are present. • `<types>` is a comma-separated list of operation types to filter the sampling by. If a sample is any of the types in `<types>`, it is recorded. Valid types are `LD` for load, `ST` for store and `B` for branch. • `<lat>` is the minimum latency. A sample is only recorded if its latency is greater than or equal to this value. The valid range is [0,4096).

Argument usage examples

- Use `--pmus-xml` and `--append-events-xml` to add support for a new PMU without having to rebuild `gatord`.

  `--pmus-xml` specifies an XML file that defines a new PMU to add to the list of PMUs that `gatord` has built-in support for. The list of built-in PMUs is defined in pmus.xml, which is in the `gatord` source directory.

  `--append-events-xml` specifies an XML file that defines one or more event counters to append to the `events.xml` file. This option allows you to add new events to `gatord` without having to rebuild `gatord` or to entirely replace `events.xml`.

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

The `events.xml` file must include the XML header and elements that are shown in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<events>
    <category name="Filesystem">
        <event counter="filesystem_loginuid" path="/proc/self/loginuid"
 title="loginuid" name="loginuid" description="loginuid"/>
    </category>
</events>
```

- The Instructions Executed counter is configured in slot 0 as:

```
--counters ARMv8_Cortex_A53_cnt0:0x08
```

To configure the cycle counter, specify `--counters <device>_ccnt`. For example:

```
--counters ARMv8_Cortex_A53_ccnt
```

Other counters do not have event codes and are identified only by name. For example:

```
--counters PERF_COUNT_SW_PAGE_FAULTS
```

Copyright © 2021–2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential