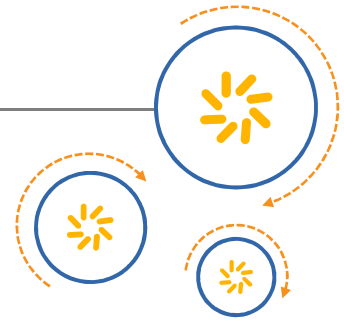




---

Qualcomm Technologies, Inc.



# Qualcomm<sup>®</sup> Snapdragon<sup>™</sup> Profiler

## User Guide

March 31, 2020

Qualcomm Snapdragon is a product of Qualcomm Technologies, Inc. Other Qualcomm products referenced herein are products of Qualcomm Technologies, Inc. or its other subsidiaries.

Qualcomm and Snapdragon are trademarks of Qualcomm Incorporated, registered in the United States and other countries. Other product and brand names may be trademarks or registered trademarks of their respective owners.

This technical data may be subject to U.S. and international export, re-export, or transfer ("export") laws. Diversion contrary to U.S. and international law is strictly prohibited.

Qualcomm Technologies, Inc.  
5775 Morehouse Drive  
San Diego, CA 92121  
U.S.A.

## Revision history

Revision	Date	Description
A	June 2016	Initial release
B	October 2016	Various updates throughout.
C	October 2017	Added sampling capture.
D	September 2018	Added Vulkan snapshot.
E	November 2019	Minor updates.
F	March 2020	Various updates throughout.

# Contents

---

<b>1 Overview</b>	<b>5</b>
1.1 Data capture modes	5
<b>2 System Requirements</b>	<b>7</b>
<b>3 Install, Launch, and Connect</b>	<b>8</b>
3.1 Install Snapdragon Profiler	8
3.1.1 Microsoft Windows installation	8
3.1.2 Mac OS X installation	8
3.1.3 Ubuntu Linux installation	8
3.2 Launch Snapdragon Profiler	9
3.3 Connect to a device	10
<b>4 User Interface</b>	<b>13</b>
4.1 Start Page	13
4.2 Views	13
4.2.1 Realtime window	13
4.2.2 Trace Capture window	14
4.2.3 Snapshot Capture window	14
4.2.4 Sampling Capture window	15
4.3 Navigation	15
<b>5 Data Capture Modes</b>	<b>16</b>
5.1 Realtime	16
5.1.1 Realtime basics	18
5.2 Trace Capture	23
5.2.1 Additional requirements	23
5.2.2 Launch applications	23
5.2.3 Trace Capture basics	24
5.2.4 Measure DSP performance	27
5.2.5 Capture OpenCL applications	27
5.2.6 Capture Vulkan applications	29
5.2.7 Analytics	30
5.2.8 Measuring time	32
5.2.9 Annotations	33
5.3 Snapshot Capture	33
5.3.1 Additional requirements	34
5.3.2 Launch applications	34
5.3.3 Snapshot Capture basics	35
5.3.4 Vulkan Snapshot Capture	37
5.4 Sampling Capture	39
5.4.1 Additional requirements	39
5.4.2 Sampling Capture basics	40
<b>6 Saving Captures</b>	<b>43</b>
<b>7 Troubleshooting</b>	<b>44</b>

## Figures

Figure 4-1 Realtime window.....	13
Figure 4-2 Trace Capture window .....	14
Figure 4-3 Snapshot Capture window .....	14
Figure 4-4 Sampling Capture window .....	15
Figure 5-1 Realtime mode.....	17
Figure 5-2 Trace Capture mode .....	23
Figure 5-3 Launch Application window.....	30
Figure 5-4 Analytics window .....	31
Figure 5-5 Measuring time with the calipers.....	32
Figure 5-6 Snapshot Capture mode .....	34
Figure 5-7 Sampling Capture mode .....	39

## Tables

Table 2-1 System requirements .....	7
Table 4-1 Navigation actions.....	15

# 1 Overview

---

Qualcomm® Snapdragon™ Profiler is system profiling software for Windows, Macintosh, and Linux computers.

Snapdragon Profiler allows developers to analyze CPU, GPU, DSP\*, memory, power, thermal, and network data, so they can find and fix performance bottlenecks.

- GPU APIs: OpenGL ES 3.1, Open CL 2.1, and Vulkan 1.0\*\*

*\* Requires a Snapdragon 820 (or later) processor*

*\*\* Requires Android N (or Android 6.0 device with a graphics driver that supports Vulkan)*

## 1.1 Data capture modes

Snapdragon Profiler provides four data capture modes offering different views of device performance: Realtime, Trace Capture, Snapshot Capture, and Sampling Capture.

### Realtime

Realtime view makes it easy to correlate system resource usage on a timeline.

- Analyze CPU, GPU, DSP\*, memory, power, thermal, and network data metrics
- Select from over 150 different hardware performance counters in 22 categories

*\* Requires a Snapdragon 820 (or later) processor*

### Trace Capture

Trace Capture mode allows you to visualize kernel and system events on a timeline to analyze low-level system events across the CPU, GPU, and DSP. View CPU scheduling and GPU stage data to see where your application is spending its time.

### Snapshot Capture

Snapshot Capture mode allows you to capture and debug a rendered frame from an OpenGL ES or Vulkan application:

- Step through and replay a rendered frame draw call-by-draw call
- View and edit shaders and preview the results on your device (preview for OpenGL ES only)
- View and debug pixel history (OpenGL ES only)
- Capture and view GPU metrics per draw call

**NOTE:** Snapdragon Capture mode requires a mobile device with a Qualcomm Snapdragon 805 (or later) processor and Android 6.0 (or later). Snapshot of Vulkan applications requires Android 8.0 (or later).

## **Sampling Capture**

Sampling Capture mode allows you to record the call graph for an application to analyze consumed CPU time. The call graph is visualized as a flame graph.

## 2 System Requirements

---

Table 2-1 lists the Snapdragon Profiler system requirements.

**Table 2-1 System requirements**

<b>Hardware</b>	<ul style="list-style-type: none"> <li>▪ PC running Microsoft Windows 7, 8.x, or 10</li> <li>▪ Mac running macOS 10.10 (Yosemite) – 10.14 (Mojave). <b>MacOS 10.15 Catalina is not yet supported.</b></li> <li>▪ PC running Ubuntu Linux 16.04 (Xenial Xerus) or later</li> </ul>
<b>Software</b>	<ul style="list-style-type: none"> <li>▪ Windows PC           <ul style="list-style-type: none"> <li>▫ Android Debug Bridge (ADB) 1.0.40 or later.</li> <li>▫ GTK# 2.12.25 for .NET at <a href="http://www.mono-project.com/download">http://www.mono-project.com/download</a></li> <li>▫ Microsoft .NET Framework 4.5 at <a href="https://www.microsoft.com/en-us/download/details.aspx?id=42643">https://www.microsoft.com/en-us/download/details.aspx?id=42643</a></li> </ul> </li> <li>▪ Mac OS X           <ul style="list-style-type: none"> <li>▫ Android Debug Bridge (ADB) 1.0.40 or later</li> <li>▫ Latest Mono Framework at <a href="http://www.mono-project.com/download">http://www.mono-project.com/download</a></li> </ul> </li> <li>▪ Linux Ubuntu           <ul style="list-style-type: none"> <li>▫ Android Debug Bridge (ADB) 1.0.40 or later</li> <li>▫ Latest Mono Framework at <a href="http://www.mono-project.com/download">http://www.mono-project.com/download</a></li> <li>▫ Latest libc++ C++ Runtime</li> <li>▫ Java JRE 1.7.0_79 or later</li> </ul> </li> </ul>
<b>Devices</b>	<ul style="list-style-type: none"> <li>▪ Any Android 5.0 (or later) device powered by a Snapdragon processor; a Nexus device is recommended; Snapdragon Capture mode requires a mobile device with a Qualcomm Snapdragon 805, (or later) processor.</li> <li>▪ Different device configurations have different metrics available in Snapdragon Profiler.</li> </ul>

**NOTE:** Snapdragon Profiler is designed to work best with devices powered by Snapdragon processors. Although some limited functionality may exist on non-Snapdragon devices, not all profiling metrics will be available.

## 3 Install, Launch, and Connect

---

### 3.1 Install Snapdragon Profiler

#### 3.1.1 Microsoft Windows installation

1. Run the Snapdragon Profiler installer on the host machine. It is recommended to run the Installer with Administrator privileges.
2. The Installer displays the required steps and prompts for any dependencies it cannot find.

#### 3.1.2 Mac OS X installation

1. Download and install the latest **Mono Framework for Mac OS X** at:  
<http://www.mono-project.com/download>
2. Once the Mono Framework is installed, click the **Snapdragon Profiler .dmg image** in the *Finder* to mount it.
3. Drag and drop the **SnapdragonProfiler.app** to the **Applications** folder.

#### 3.1.3 Ubuntu Linux installation

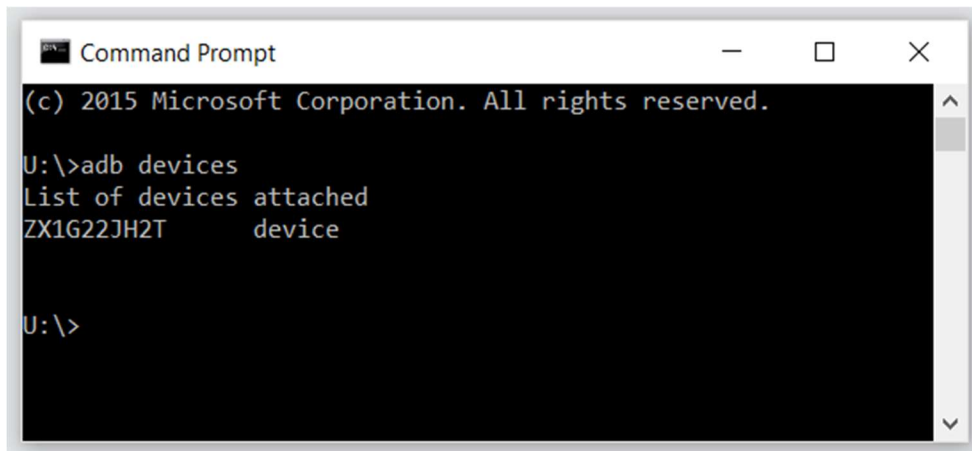
1. Download and install the latest **Mono Framework for Ubuntu Linux** at:  
<http://www.mono-project.com/download> - 'mono-complete' is required.
2. Install the libc++ C++ Standard library with `sudo apt-get install libc++1 libc++abi1`.
3. Install Java Runtime 1.7.0\_79 or later with `sudo apt-get install default-jre`.
4. Download the Android SDK and set up the paths accordingly.
5. Extract the **Snapdragon Profiler tarball** to the preferred directory with `tar zxvf SnapdragonProfiler_release_External_Linux.tar.gz`.



## 3.2 Launch Snapdragon Profiler

**NOTE:** Start with Step 5 if the mobile device is already connected to a computer and communicating via ADB.

1. Connect the Android device to a computer where Snapdragon Profiler is installed.
2. Choose **Android Settings > Developer Options** to ensure the device has Developer Options enabled.
  - a. If **Developer Options** is not visible, go to **Android Settings > About phone > Software info**, and continuously tap the Build number until **Developer Options** is enabled.
  - b. Go back one step to **Android Settings** to confirm that the **Developer Options** menu item is available.
3. If you don't have ADB in your path, in Snapdragon Profiler go to **File > Settings** and set the ADB path on the Android tab.
4. On a Windows PC (or terminal on OS X or Linux), open a **command prompt** and run the **adb devices** command to confirm that the device is recognized (appears in the **List of devices attached**).
  - a. If ADB does not recognize the device, confirm that the USB connection is in place or ADB has been set up over Wi-Fi. Also verify that the latest ADB USB drivers for the device are installed.
  - b. If you have problems connecting, confirm that ADB 1.0.32 (or later) is installed. You can check this by typing 'adb version' in the cmd prompt.
  - c. If ADB recognizes the device, but it shows as **Unauthorized**, authorize the computer through a pop-up authorization window on the device.



```
Command Prompt
(c) 2015 Microsoft Corporation. All rights reserved.

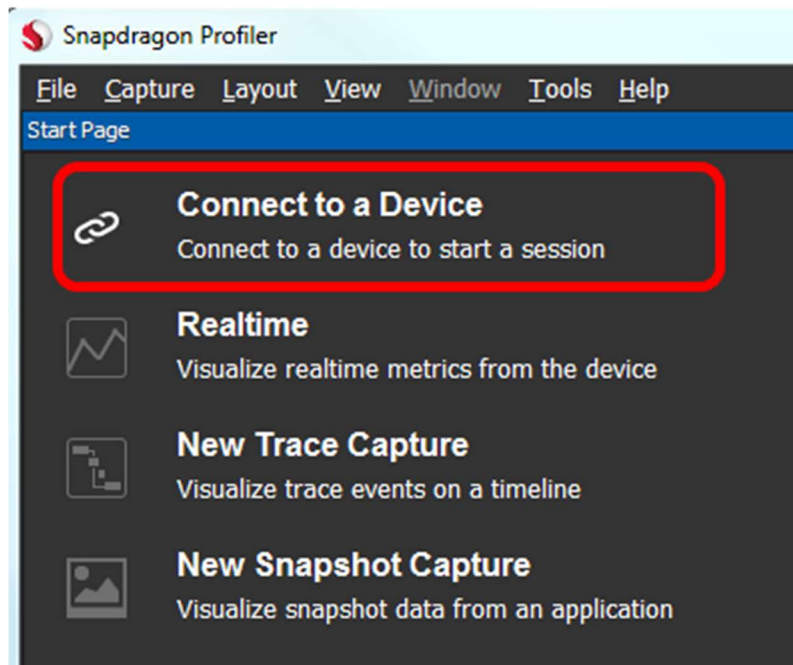
U:\>adb devices
List of devices attached
ZX1G22JH2T    device

U:\>
```

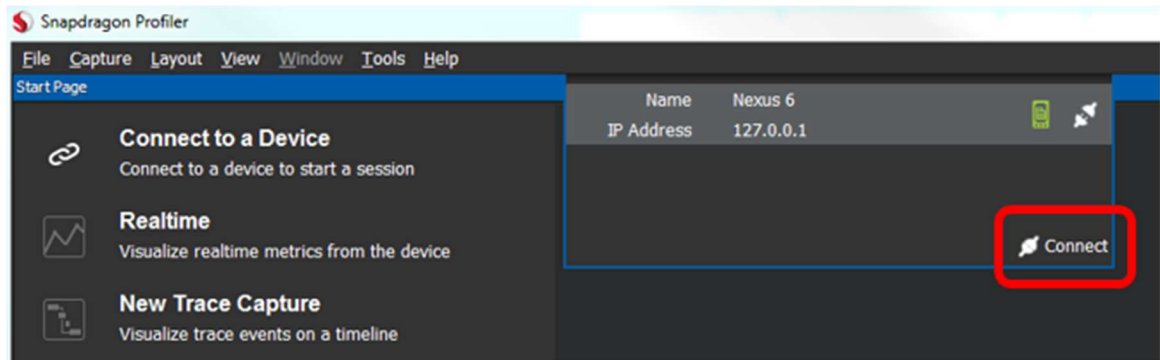
5. Launch Snapdragon Profiler:
  - a. For Windows, go to **Start > Programs > Qualcomm > Snapdragon Profiler** or double-click the Snapdragon Profiler desktop icon.
  - b. For Mac OS X, go to **Finder > Applications** folder, and double-click **Snapdragon Profiler**.
  - c. For Ubuntu Linux, execute the **run\_sdp.sh** script from the root Snapdragon Profiler directory.
6. Click **Yes** if asked whether to allow Snapdragon Profiler to send anonymous user statistics.

### 3.3 Connect to a device

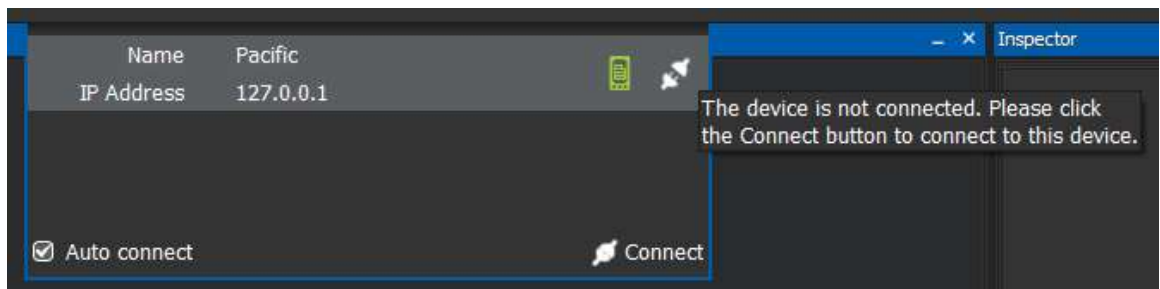
1. After launching Snapdragon Profiler, go to **File > Connect**, or from the *Start Page* click the icon to the left of **Connect to a Device** to connect to a device to start a session.



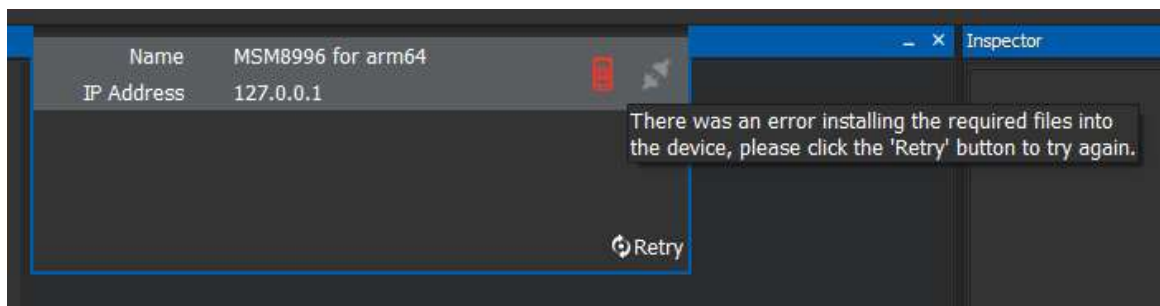
- The *Connection* window slides from the top of the main window. Snapdragon Profiler auto-detects any devices connected to the computer via ADB, over USB or Wi-Fi, and begins installing the service on the device.



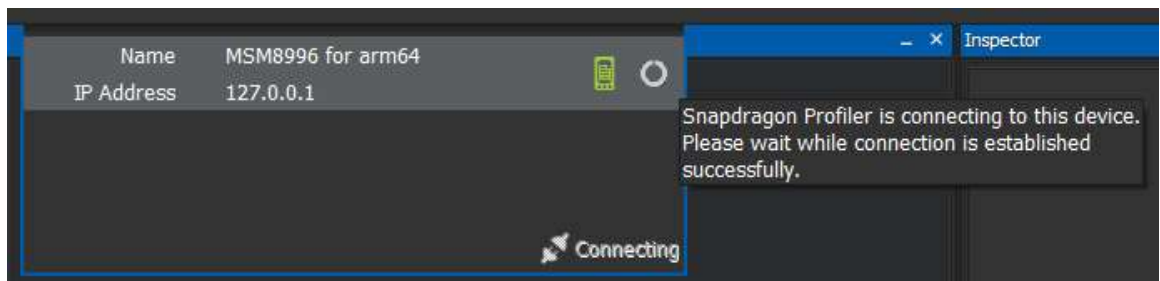
- If the installation is successful, the *Device* icon is **green** and the Connect button is enabled. Check the Auto Connect box to automatically connect if a single device is detected.



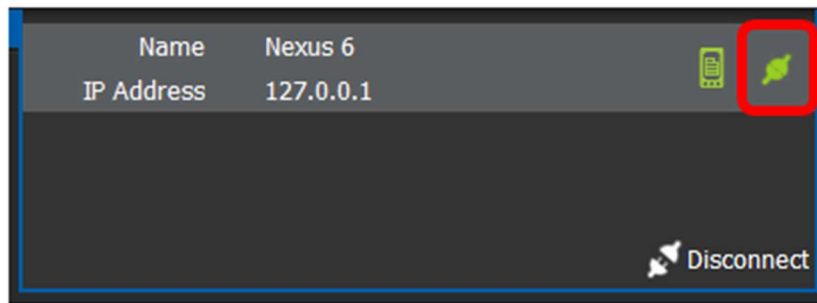
If the installation fails, the *Device* icon is **red** and the Retry button is enabled. Click **Retry** to try installing the service again.



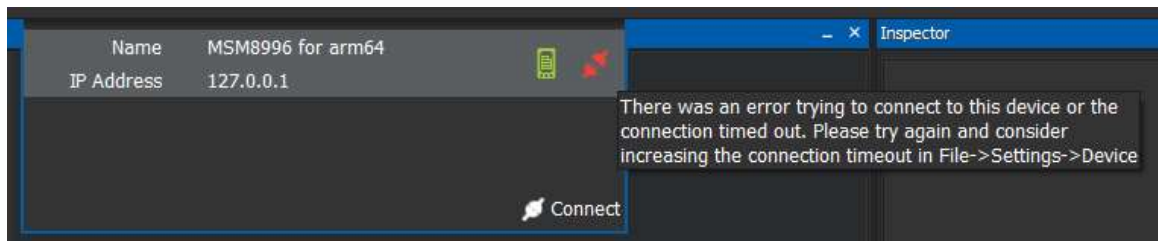
- Click **Connect** to connect to the device. A progress wheel displays while Snapdragon Profiler is connecting to the device:



5. If the connection is successful, the *Connection* icon is **green**:



If you receive a connection error (the *Connection* icon is **red**), click **Connect** to try again.



**NOTE:** Mouse over the **red** *Connection* icon to get specific details about the error.

6. Once connected, Snapdragon Profiler can be used in four data capture modes: Realtime, Trace Capture, Snapshot Capture, and Sampling Capture. See Chapter 5 for details on each of these modes.

**NOTE:** If you have problems connecting, make sure you have ADB 1.0.40 (or later) installed.

# 4 User Interface

Snapdragon Profiler provides pre-defined metrics and layouts for one-click setup that lets users quickly capture and view profiled data.

Snapdragon Profiler also enables user-defined metrics and dockable views for customization of the user interface.

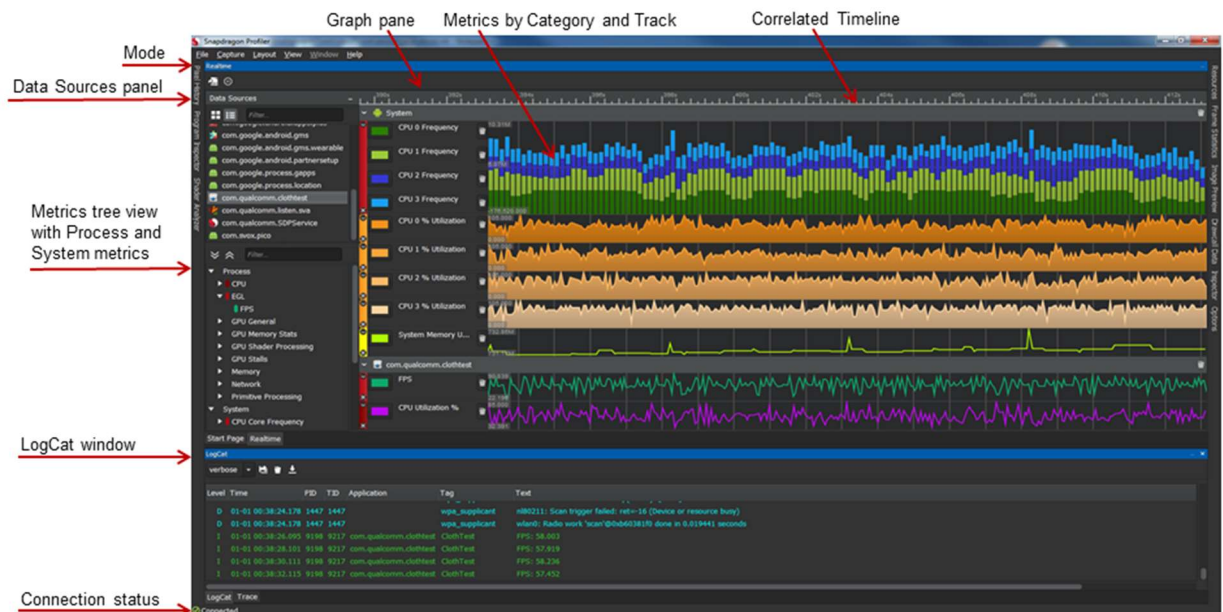
## 4.1 Start Page

Similar to Visual Studio, the Snapdragon Profiler **Start Page** provides easy access to the primary Profiler functions:

- Connect to a Device
- Realtime profiling
- New Trace Capture
- New Snapshot Capture

## 4.2 Views

### 4.2.1 Realtime window



**Figure 4-1 Realtime window**

NOTE: The LogCat window is now hidden by default.

## 4.2.2 Trace Capture window

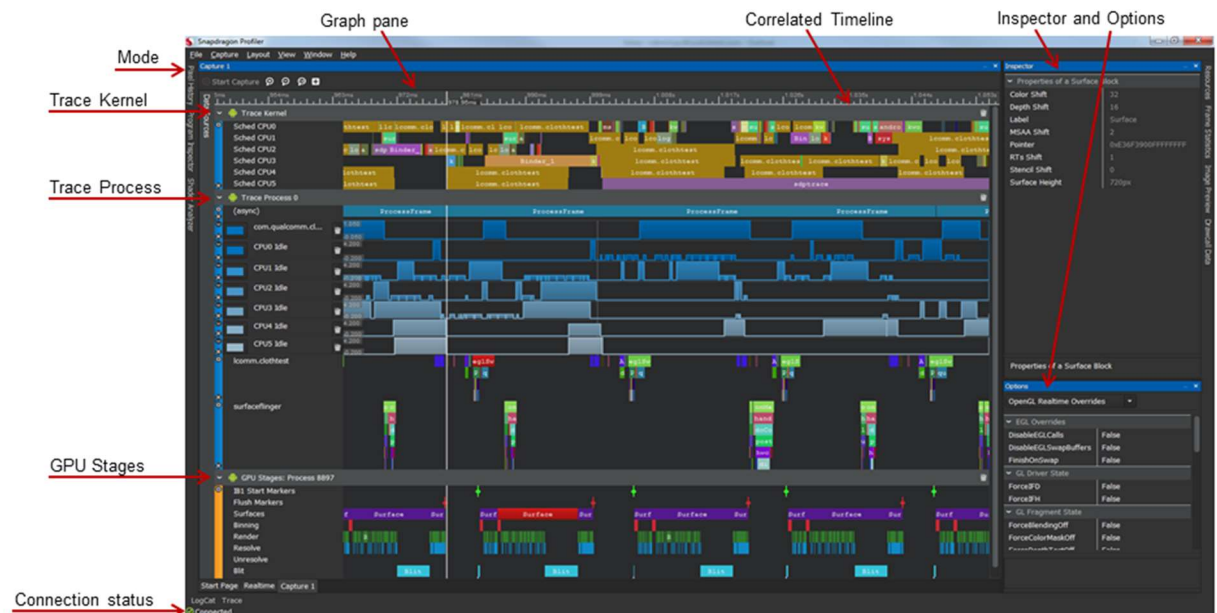


Figure 4-2 Trace Capture window

## 4.2.3 Snapshot Capture window

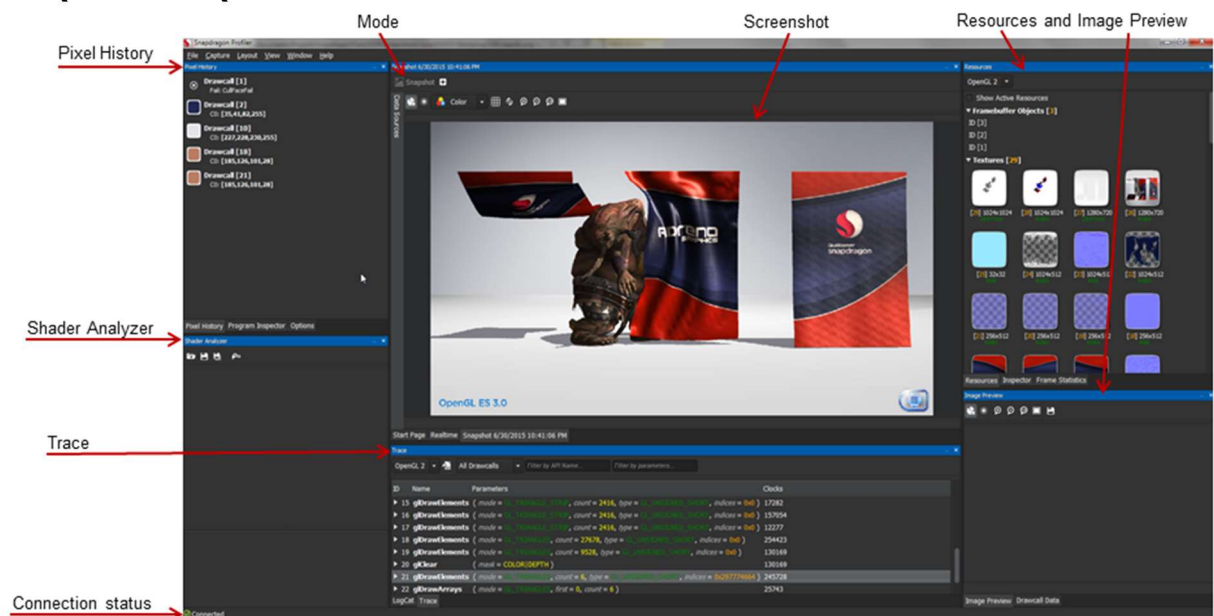


Figure 4-3 Snapshot Capture window



## 4.2.4 Sampling Capture window

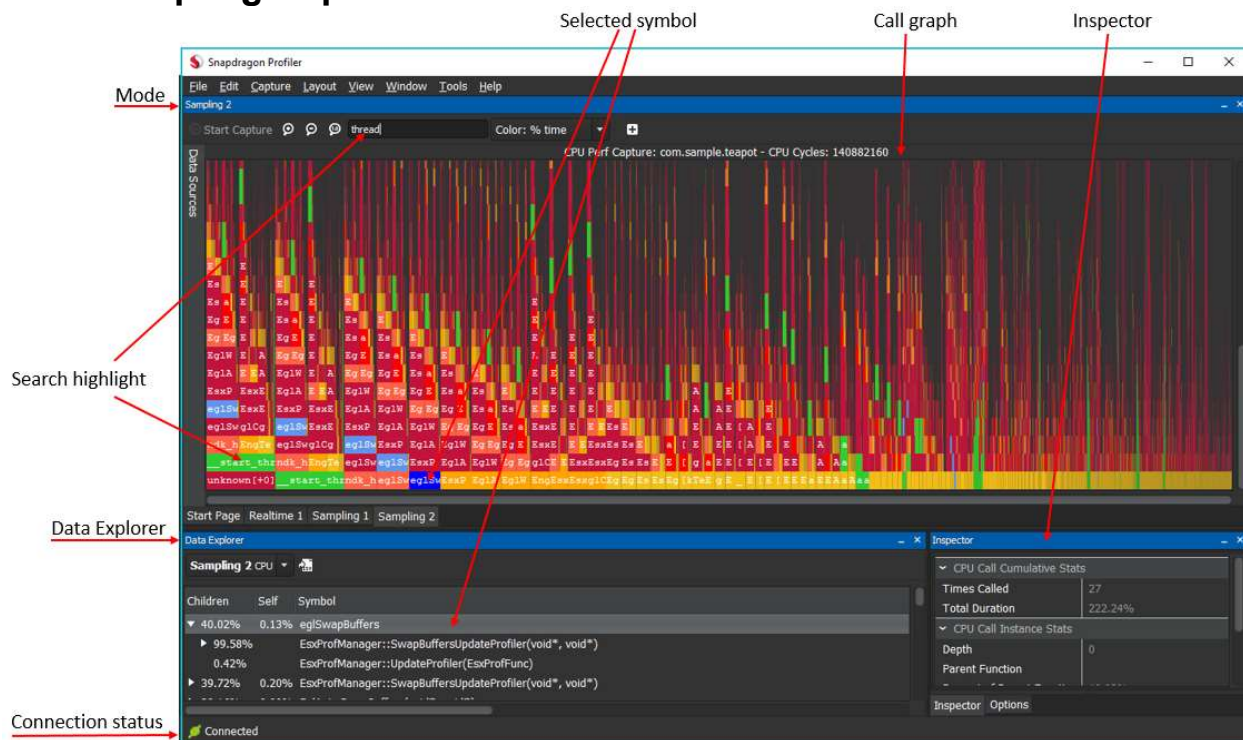


Figure 4-4 Sampling Capture window

## 4.3 Navigation

Table 4-1 describes Snapdragon Profiler user interface navigation.

Table 4-1 Navigation actions

Navigation Action	Purpose/Result
Drag and drop	To select a category or individual metric to view in the Graph pane To rearrange categories and tracks in the Graph pane
Turn the mouse wheel	To zoom in/out
Left-click, hold, and move the mouse	To pan
Click the Timeline	To visualize the time correlation across elements
Double-click System Metric heading	To add all System metrics to the graph pane and begin profiling
Ctrl + W	To zoom in on Realtime or Trace Capture mode
Ctrl + S	To zoom out on Realtime or Trace Capture mode
Ctrl + A	To pan left when paused in Realtime or Trace Capture mode
Ctrl + D	To pan right when paused in Realtime or Trace Capture mode

# 5 Data Capture Modes

---

This chapter provides an overview of the Realtime, Trace Capture, and Snapshot Capture modes including features and functionality.

## 5.1 Realtime

Realtime mode, shown in [Figure 5-1](#), allows streaming and viewing real-time performance metrics while an application is running on the Snapdragon-powered device.

Realtime mode plots real-time GPU and system performance data streaming from the embedded graphics driver, including system and process metrics.

### System Metrics (categories)

- CPU Core Frequency
- CPU Core Load
- CPU Core Utilization
- Additional CPU metrics may be available for some devices.
- GPU General
- GPU Memory Stats
- GPU Shader Processing
- GPU Stalls
- Network - Cellular
- Network – Wi-Fi
- Power
- Primitive Processing
- System Memory
- Thermal



## Process Metrics (categories)

- CPU
- EGL (not available for every process)
- GPU General (not available for every process)
- GPU Memory Stats (not available for every process)
- GPU Shader Processing (not available for every process)
- GPU Stalls (not available for every process)
- Memory
- Network
- Primitive Processing (not available for every process)

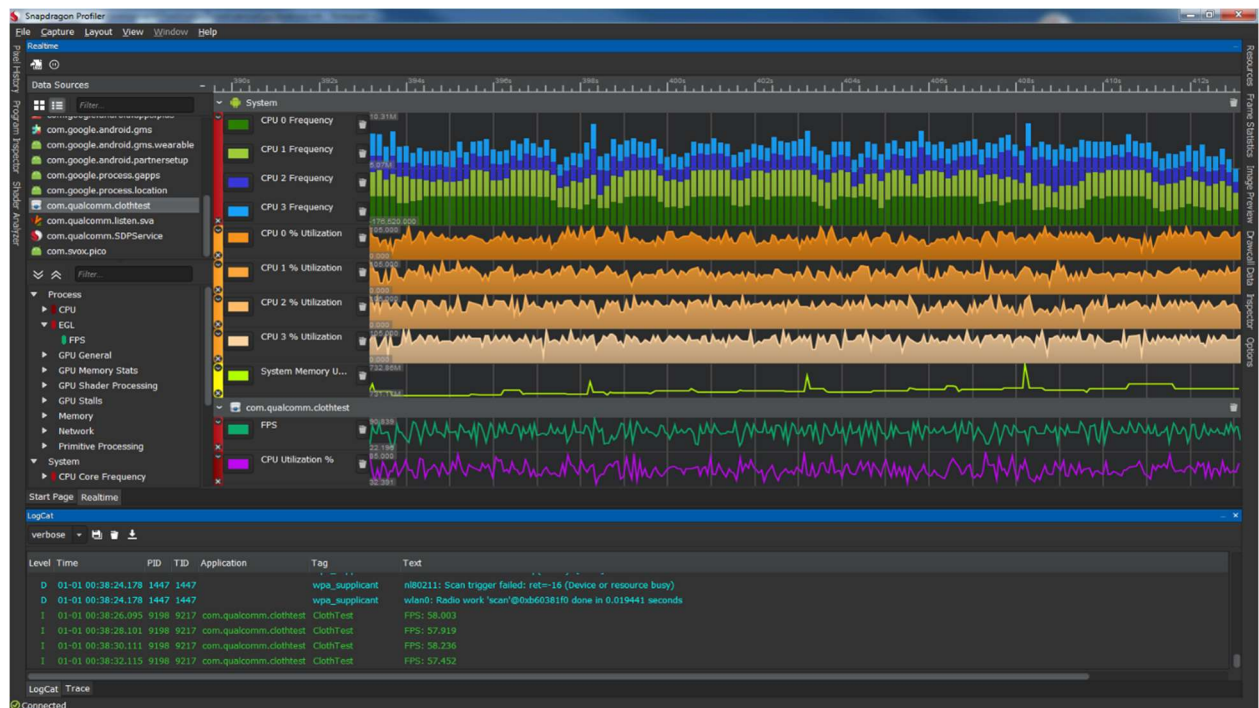
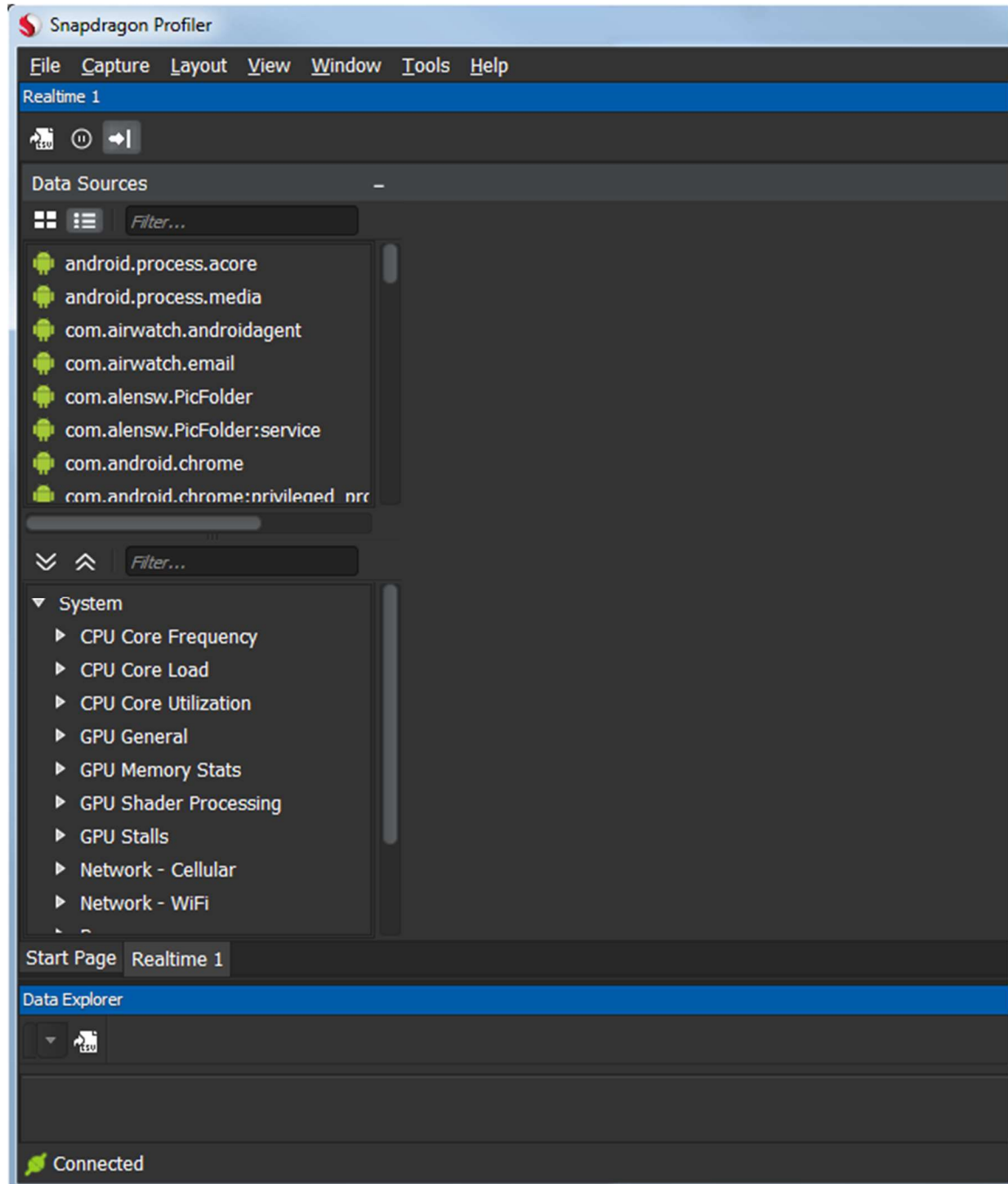


Figure 5-1 Realtime mode

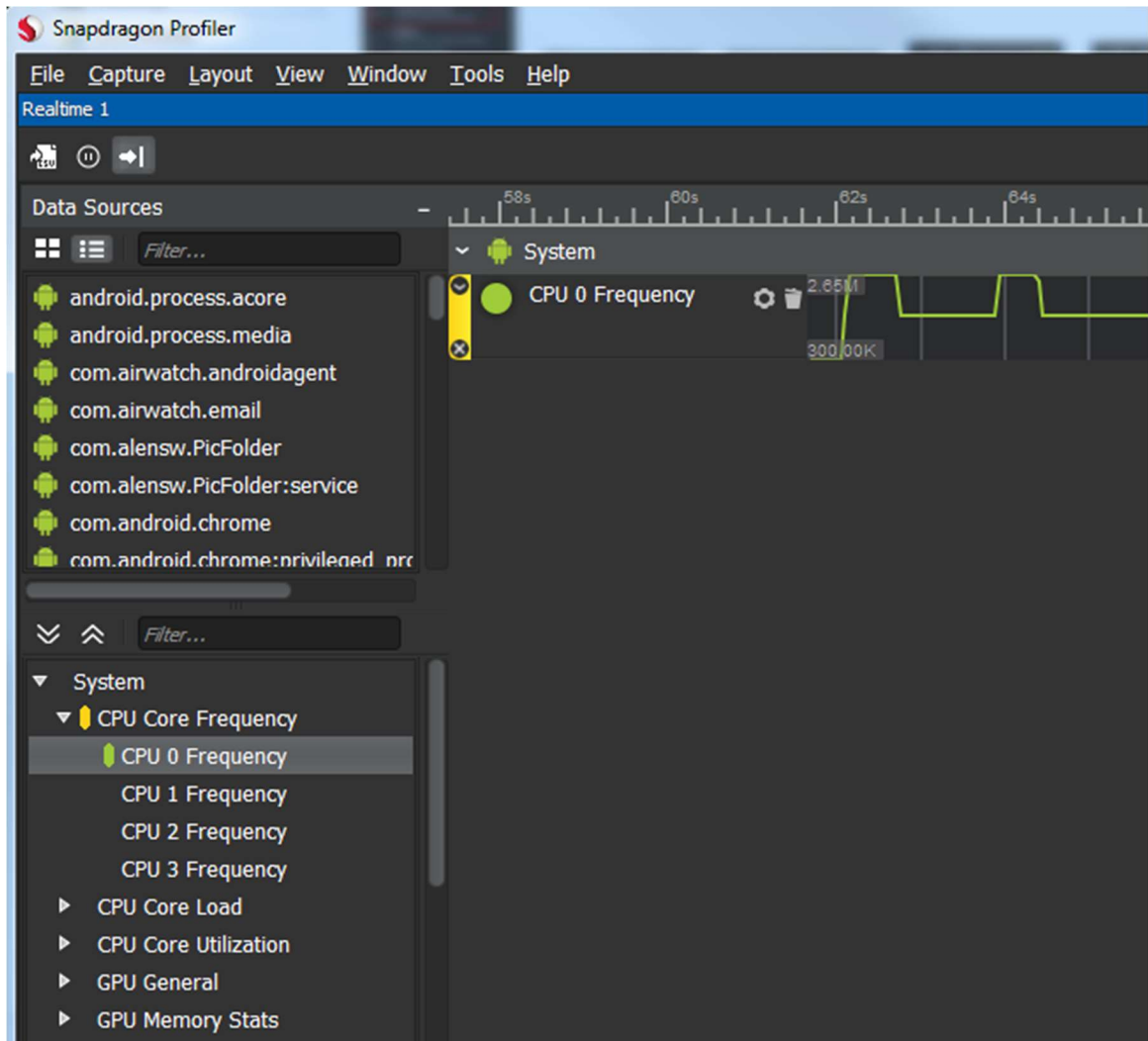
### 5.1.1 Realtime basics

1. Check that the Android device is connected to the computer where Snapdragon Profiler is installed and launch the Profiler (see Section 3.2).
2. Choose **File > Connect**, or from the *Start Page*, click the icon to the left of **Connect to a Device** to connect to a device to start a session (see Section 3.3).
3. From the **Start Page**, click **Realtime**, or click the **Realtime** tab to see real-time metrics for an application or device. Real-time metrics can be gathered for CPU, GPU, memory, network, thermal, and power as shown in this view of the **Realtime** tab:



- On the left of the **Realtime** tab, **System** metrics can be chosen from the **Metrics** tree view in the **Data Sources** panel.

To add a new metric graph, double-click the category (to add all the metrics in a category) or individual metric, or drag-and-drop the category or metric into the **Graph** pane on the right.

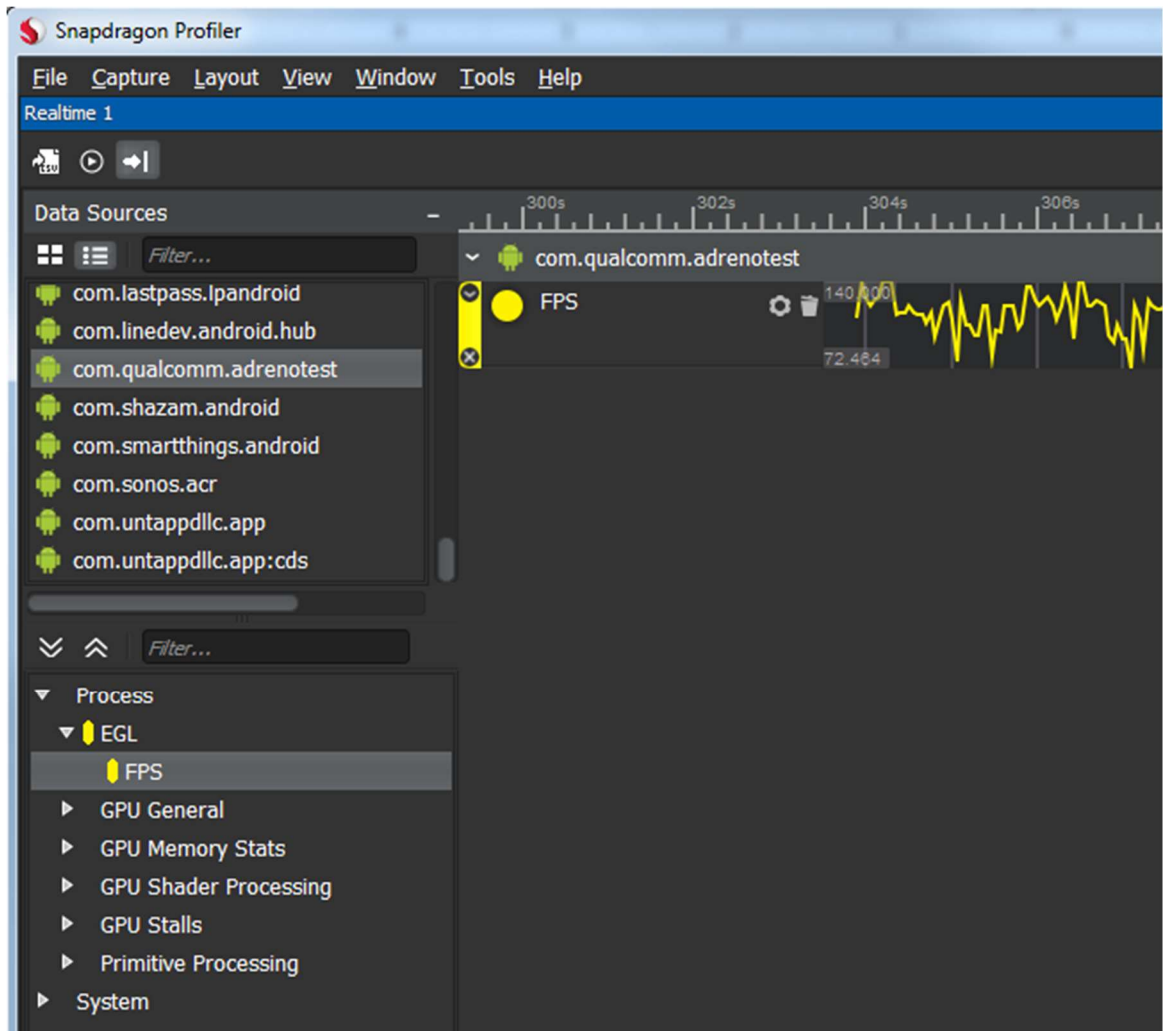


- Application, or per-process, metrics can be viewed by first clicking on a process in the **Process** list shown in the top area of the **Data Sources** panel. The **Process** list shows all active user processes.

**NOTE:** Not all metrics are available for every process.

6. Select a process to see all the metrics available for that process in the **Metrics** list.

Adding process metrics to the **Graph** pane is the same as with system metrics – double-click, or drag-and-drop, the category or individual metric into the pane.

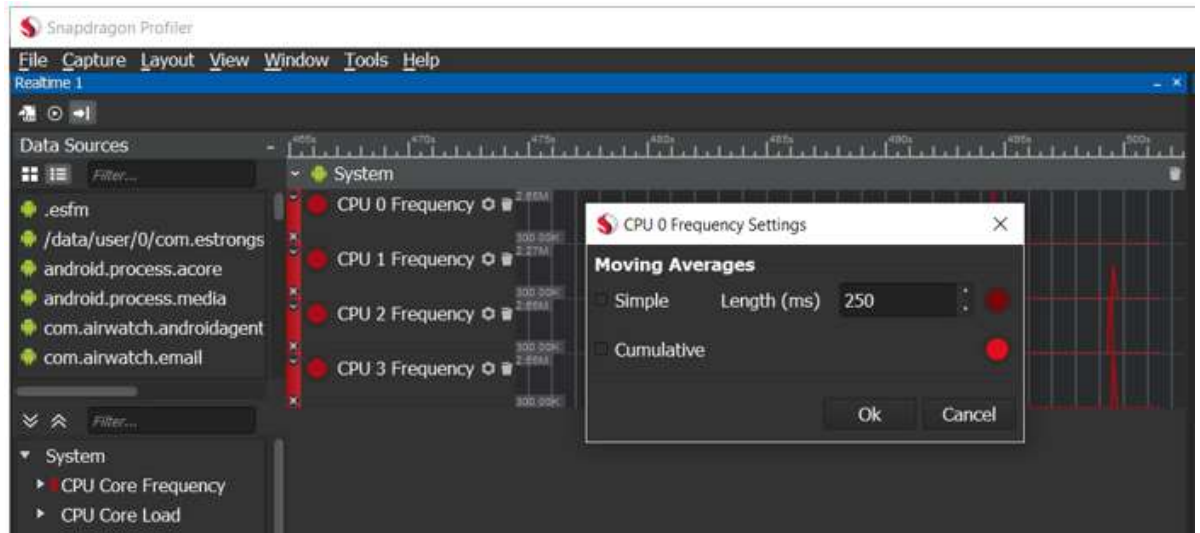


7. System and process metrics can be viewed in the **Graph** pane at the same time. Multiple processes can also be viewed.



NOTE: The more metrics selected and displayed, the more intrusive Snapdragon Profiler is on application performance.

8. If desired, click the gear icon in the Graph pane and select **Simple** and/or **Cumulative** to display a moving average.





## 5.2 Trace Capture

Trace Capture mode, shown in Figure 5-2, captures trace events and data available on Snapdragon-powered devices. Once captured, the call trace can be used to:

- Collect and display important performance statistics
- See how threads are scheduled on each of the CPU cores in the Kernel
- See the correlation between CPU, DSP, and GPU activity on the same timeline

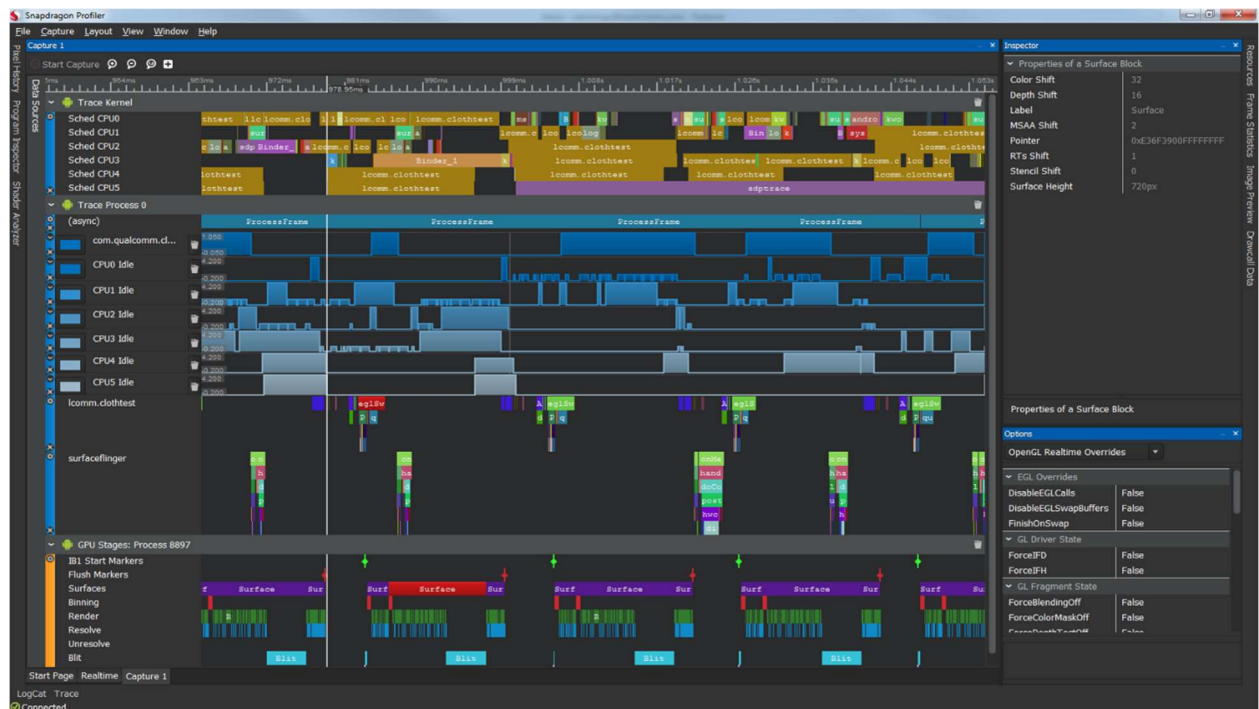


Figure 5-2 Trace Capture mode

### 5.2.1 Additional requirements

OpenGL ES, Vulkan, and OpenCL apps must include `android.permission.INTERNET` in the app's `AndroidManifest.xml` to enable API tracing and GPU metrics.

### 5.2.2 Launch applications

The **Launch Application** button in the **Trace Capture** panel can be used to launch any application installed on the target device. The implications of using this button depend on the type of app being launched.

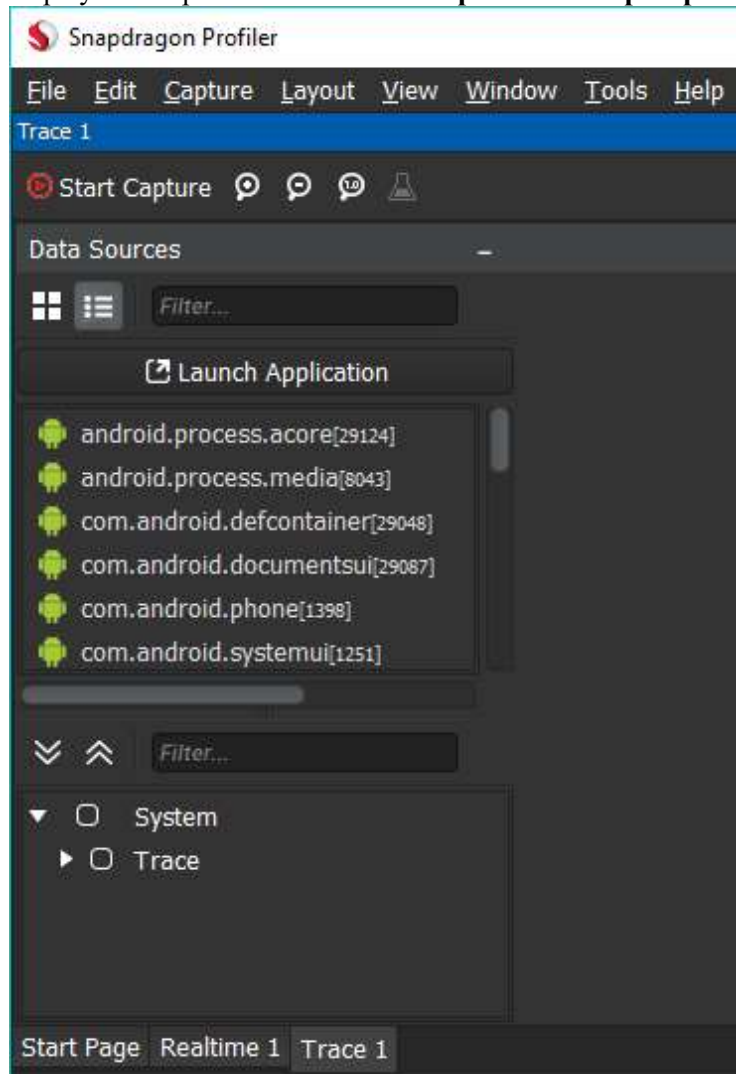
- Vulkan apps *must* be launched using the button to enable capture of Vulkan trace metrics.
- Using the button for OpenGL ES apps will allow Snapdragon Profiler to load the most current version of its OpenGL API interceptor library, which will enable optimal profiling functionality. When OpenGL ES apps are launched on the device without using the button, the API interceptor library that is pre-installed on the device will be used. Depending on the age of that library it may lack functionality or contain bugs that have been fixed in the current version.

- All other apps behave the same whether they are launched from the device or with the button.

Launching an app using the **Launch Application** button makes use of the Java Debug Wire Protocol. Therefore, it is recommended to disable other debugging and profiling tools which use JDWP when using this functionality.

### 5.2.3 Trace Capture basics

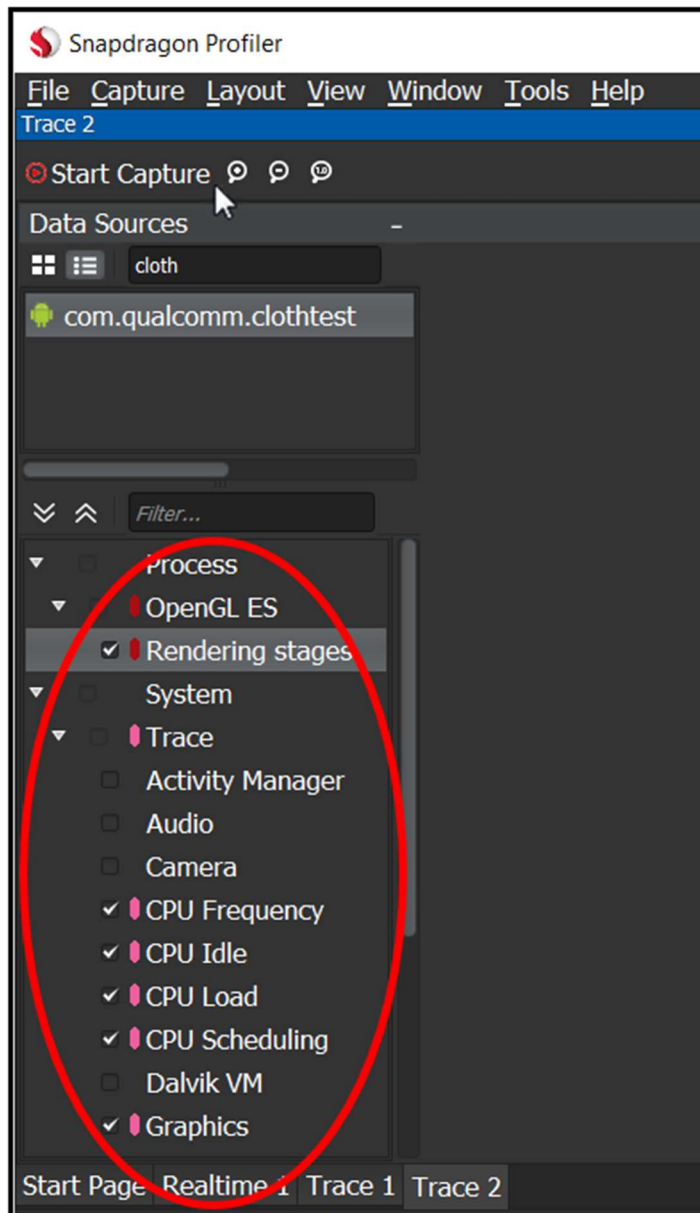
1. Check that the Android device is connected to the computer where Snapdragon Profiler is installed and launch the Profiler (see Section 3.2).
2. Choose **File > Connect**, or from the *Start Page*, click **Connect to a Device** to connect to a device to start a session (see Section 3.3).
3. From the *Start Page*, click **New Trace Capture**, or choose **Capture > New Trace**
4. A new **Capture** tab is created. When this tab is selected, a view similar to **Realtime** is displayed except traces have **Start Capture** and **Stop Capture** buttons.



5. From the **Data Sources** panel, choose the metrics to view. Choose a process from the **Process** list to view metrics specific to that process, if any exist.

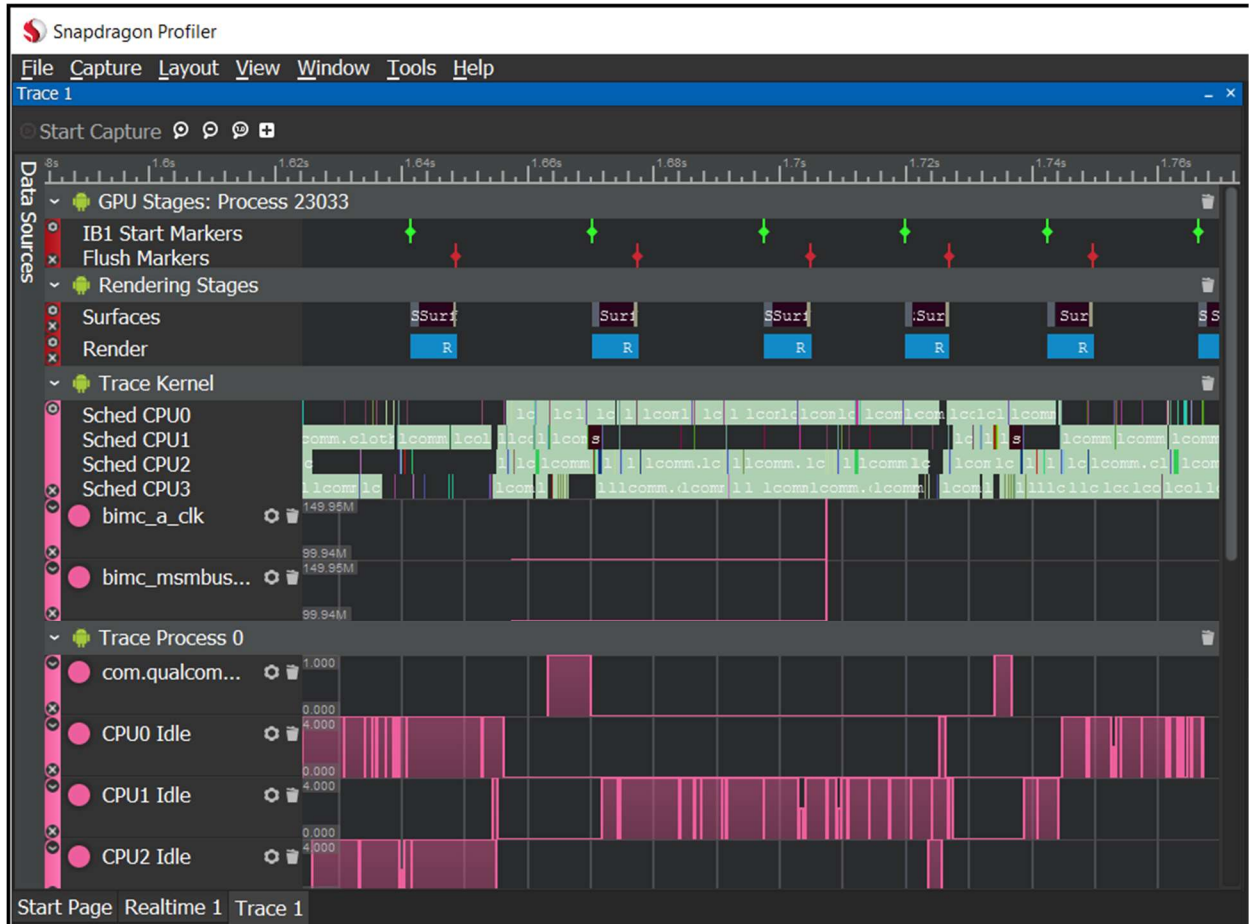


6. Click **Start Capture** to start the capture.



- Click **Stop Capture** to stop the capture after a few seconds. Snapdragon Profiler processes and displays the trace data in the **Graph** pane.

**NOTE:** Snapdragon Profiler does not allow trace captures longer than 10 seconds.



## 5.2.4 Measure DSP performance

To collect DSP data, a device with a Snapdragon 820 processor such as a Samsung Galaxy S7 must be used.

1. Check that the Android device is connected to the computer where Snapdragon Profiler is installed and launch the Profiler (see Section 3.2).
2. Choose **File > Connect**, or from the **Start Page** click **Connect to a Device** to connect to a device to start a session (see Section 3.3).
3. From the **Start Page**, click **New Trace Capture**, or choose **Capture > New Trace** to view trace-level performance data for the application or device.
4. From the **Metrics** window, expand the **System** category if needed, two subcategories are displayed: Trace and DSP. Trace collects data from the Android system trace and DSP collects data from the device DSP.
5. Select the metrics for the capture:
  - a. Expand the DSP subcategory to view the available metrics. Scroll to DSP % Utilization and check the box to enable this metric.
  - b. Expand the Trace subcategory and enable the CPU Frequency and HW Modules metrics.

Collecting DSP and Trace metrics will demonstrate Profiler's ability to correlate Timeline information across different processor subsystems.

**NOTE:** To see a change in the DSP % Utilization, an app that produces audio needs to be turned on during the capture.

6. Click **Start Capture** in the top left corner to start the capture.
7. A few seconds into the capture, turn on the audio on.
8. Let the audio play for a few seconds and then click **Stop Capture**. The capture can be continued, however it automatically stops capturing after 10 seconds.
9. The capture data displays in the main Profiler window.
  - Notice that a few seconds into the capture, the DSP % Utilization increased due to the audio activity.
  - Notice the activity increase in the Trace data during the time the audio was running.
10. Use the bookmark feature to easily see correlation across timelines.

## 5.2.5 Capture OpenCL applications

Use Trace Capture to understand how an OpenCL application performs on a device powered by a Snapdragon processor.

**NOTE:** Not all mobile devices support OpenCL. Use an app like OpenCL Info to see if OpenCL is supported on your device. Depending on the application being captured and the preferred behavior, additional setup steps may be required before running an app.

1. Check that the Android device is connected to the computer where Snapdragon Profiler is installed and launch the Profiler (see Section 3.2).
2. Choose **File > Connect**, or from the *Start Page* click **Connect to a Device** to connect to a device to start a session (see Section 3.3).
3. If the application to be captured is command line, and capture from the beginning is preferred, the **Enable Blocking** option must be set before running the application. Once set, the application pauses execution until the capture begins.

NOTE: An OpenCL preset layout is available for easy access to all OpenCL-relevant views.

4. From the **Start Page**, click **New Trace Capture**, or choose **Capture > New Trace** to view trace-level performance data for an application or device.
5. From the **Process** list, choose the application. An **OpenCL Trace** metric appears in the **Metrics** tree view.
6. Enable the **OpenCL Trace** metric and the other OpenCL metrics to capture.
7. Click **Start Capture** in the top left corner to start the capture.

NOTE: If the app being captured is command line, execution resumes on the given app.

8. Click **Stop Capture** to end the capture. If the application finishes executing, the capture stops automatically.
9. The data is then transferred and processed. A progress indicator displays in the bottom right corner of the Profiler window.
10. When processing completes, the Trace Capture data displays in the main Profiler window, organized by thread on the host, and by context and device on the GPU.
  - Event links  
Correlation links are shown between calls that either wait or complete events.
  - Trace elements  
Select trace elements to view detailed information about the element in Trace view.
  - Buffer visualization  
If buffer and image data transmission was enabled, an additional track displays with markers where data bound to the given buffers might have been modified.
  - Buffer snapshots track  
Click a marker to view this data and look at the selected buffer data on the buffer or image viewer.
  - Buffer viewer  
The Buffer viewer can be customized to suit the data being viewed. The number of rows and columns per page can be modified to align the data and the display format can be changed to match the most common use cases.
  - Trace view  
Trace data with API calls return values, parameters and metric calculations appears in the Trace view. Metrics are only calculated on relevant calls.

- Resources and Inspector views

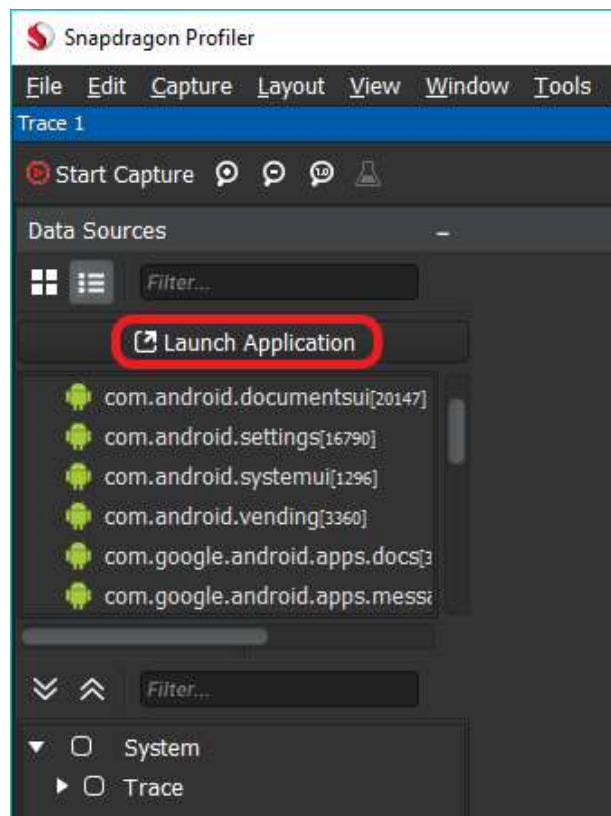
The Resources view lists all resources gathered during the capture. Click a resource to display details about the resource in Inspector view.

Programs display static code analysis within Inspector view as well as the containing kernels within the source view.

## 5.2.6 Capture Vulkan applications

To capture Vulkan trace metrics for a Vulkan application, the application *must* be launched from the **Trace Capture** panel of Snapdragon Profiler. Additionally, Vulkan trace metrics require your app to be debuggable (except on rooted devices). Set the `android:debuggable` flag to true in the app's AndroidManifest.xml.

1. Follow steps 1-4 of [Trace Capture basics](#) to connect and open a new **Trace Capture** tab, then click the **Launch Application** button in the **Data Sources** panel:



2. The **Launch Application** window is shown in [Figure 5-3](#). Select an application from the list of installed packages and click **Launch**. If the application is already running, it will restart with Vulkan trace profiling enabled.

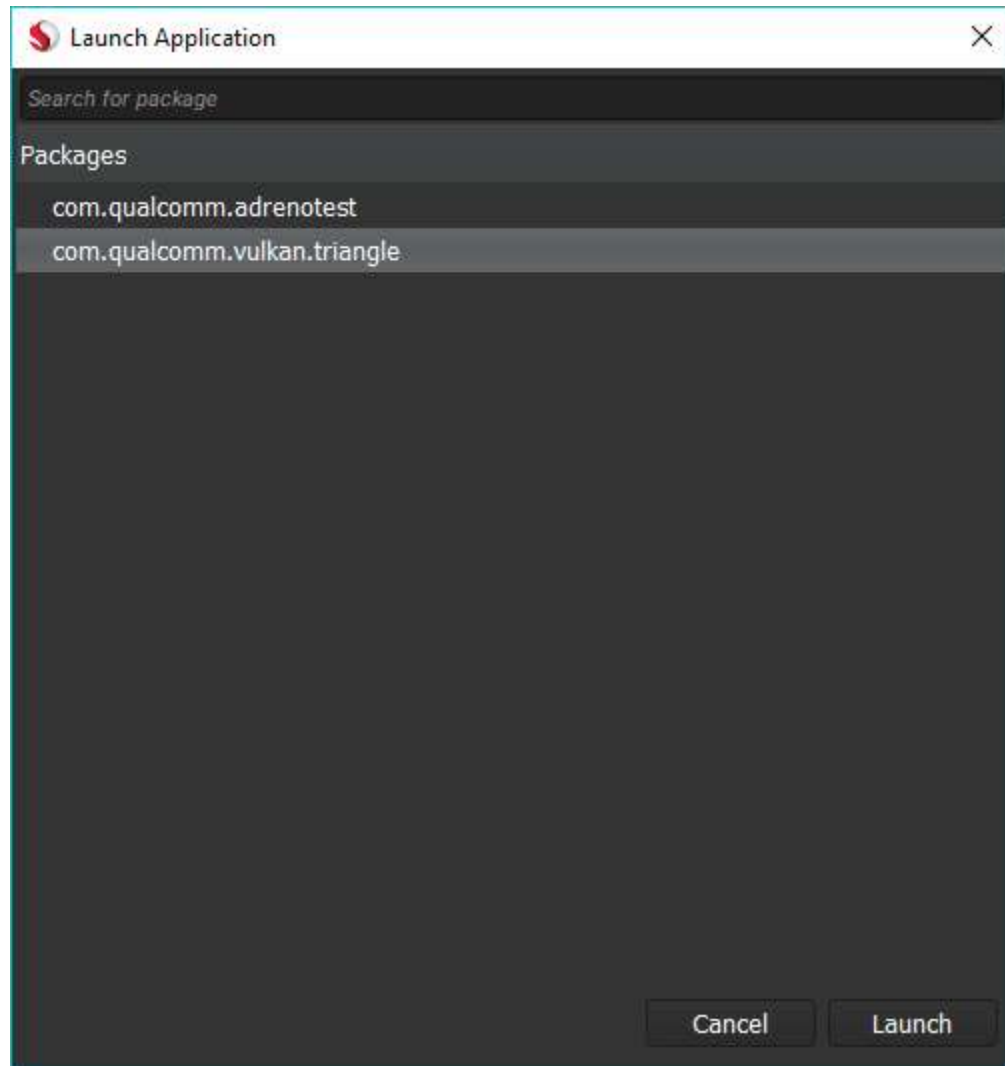
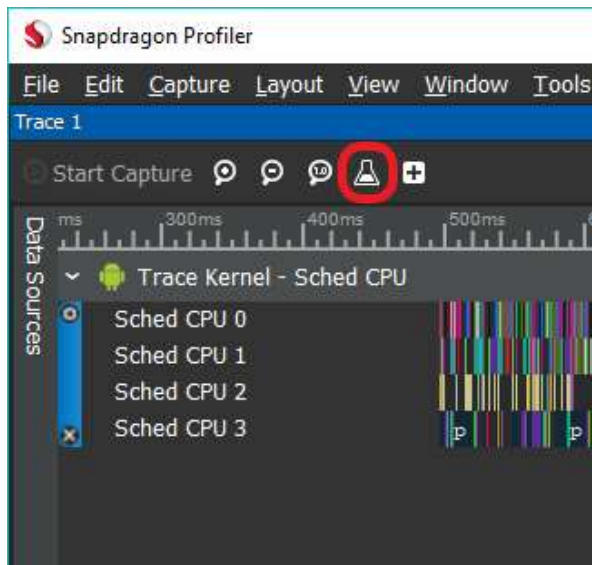


Figure 5-3 Launch Application window

3. Proceed from step 5 of [Trace Capture basics](#) to select metrics and start a capture.

## 5.2.7 Analytics

Some trace metrics support additional analytics. After a trace has been captured, click the **Analytics** button on the toolbar if it is enabled.



The **Analytics** window is shown in Figure 5-4. Select a statistic in the panel on the left to view additional information.

Statistics		Thread Time Per Core											
	Name	PID	Priority	Core 0	%	Core 1	%	Core 2	%	Core 3	%	All Cores	%
▼ Thread Level Data													
Runqueue Depth	Total	0	0	430.45ms	100.000000	485.49ms	100.000000	178.8ms	100.000000	459.62ms	100.000000	1.554s	100.000000
Thread Time Per Core	▼ Threads	0	0	0.000000		0.000000		0.000000		0.000000		0.000000	
	ksoftirqd/2	23	120	0.000000		0.000000		17.48ms	9.773435	0.000000		17.48ms	1.124255
	rcu_preempt	7	98	19.43ms	4.513701	19.62ms	4.041467	0.000000		0.000000		39.05ms	2.512285
	SnapshotPlayer3	30389	120	27.96ms	6.495603	0.000000		3.94ms	2.205245	219us	0.047648	32.12ms	2.066571
	ksoftirqd/0	3	120	22.88ms	5.315894	0.000000		0.000000		0.000000		22.88ms	1.472115
	hwservicemanager	475	120	0.000000		0.000000		1.24ms	0.692949	0.000000		1.24ms	0.079711
	configstore@1.0	4776	100	0.000000		0.000000		339us	0.189596	0.000000		339us	0.021810
	mdss_fb0	19854	83	26.1ms	6.064189	10.26ms	2.112702	9.82ms	5.491021	0.000000		46.18ms	2.970865
	kgs_l_worker_thr	244	97	19.45ms	4.519044	22.85ms	4.706566	0.000000		0.000000		42.3ms	2.721502
	kschedfreq:2	19853	49	0.000000		0.000000		6.160480	5.7ms	1.239274		16.71ms	1.075103
	kschedfreq:0	688	49	8.28ms	1.923823	14ms	2.883879	0.000000		0.000000		22.28ms	1.433514
	kworker/u8:8	23525	120	28.98ms	6.731638	0.000000		1.24ms	0.695186	0.000000		30.22ms	1.944142
	kworker/u8:2	28421	120	5.98ms	1.389492	24.22ms	4.988342	5.07ms	2.836114	0.000000		35.27ms	2.269098
	vsync_retire_wo	162	94	1.96ms	0.454181	2.54ms	0.522357	0.000000		0.000000		4.49ms	0.288929
	rcuc/2	22	98	0.000000		0.000000		2.63ms	1.468113	0.000000		2.63ms	0.168880
	rcuc/0	10	98	4.61ms	1.071449	0.000000		0.000000		0.000000		4.61ms	0.296713
	pluginGPU-QGL	30323	120	18.84ms	4.377563	41.86ms	8.622799	4.43ms	2.478733	254.07ms	55.277792	319.21ms	20.536271
	sdpsservice	30220	120	4.88ms	1.132549	0.000000		0.000000		0.000000		4.88ms	0.313634
	rcu_sched	8	98	486us	0.112906	332us	0.068384	0.000000		0.000000		818us	0.052626

Figure 5-4 Analytics window



## 5.2.8 Measuring time

The **calipers**, demonstrated in [Figure 5-5](#), measure time between two markers.

1. Double click on the timeline (see green box) to set markers at the beginning and end of the region you wish to measure
2. Left click one of the markers, then shift + left click on the other

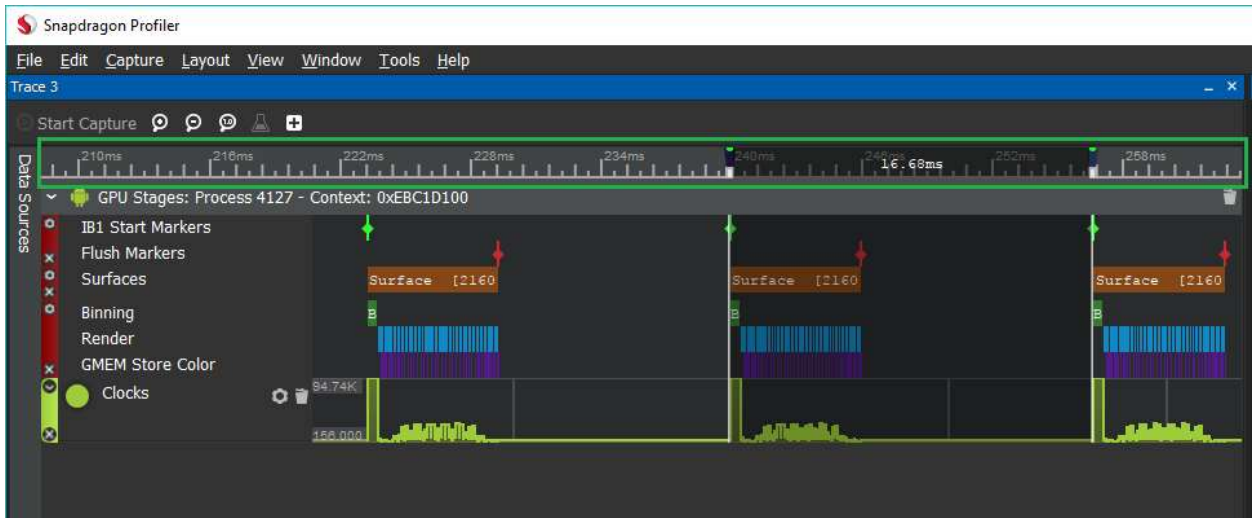
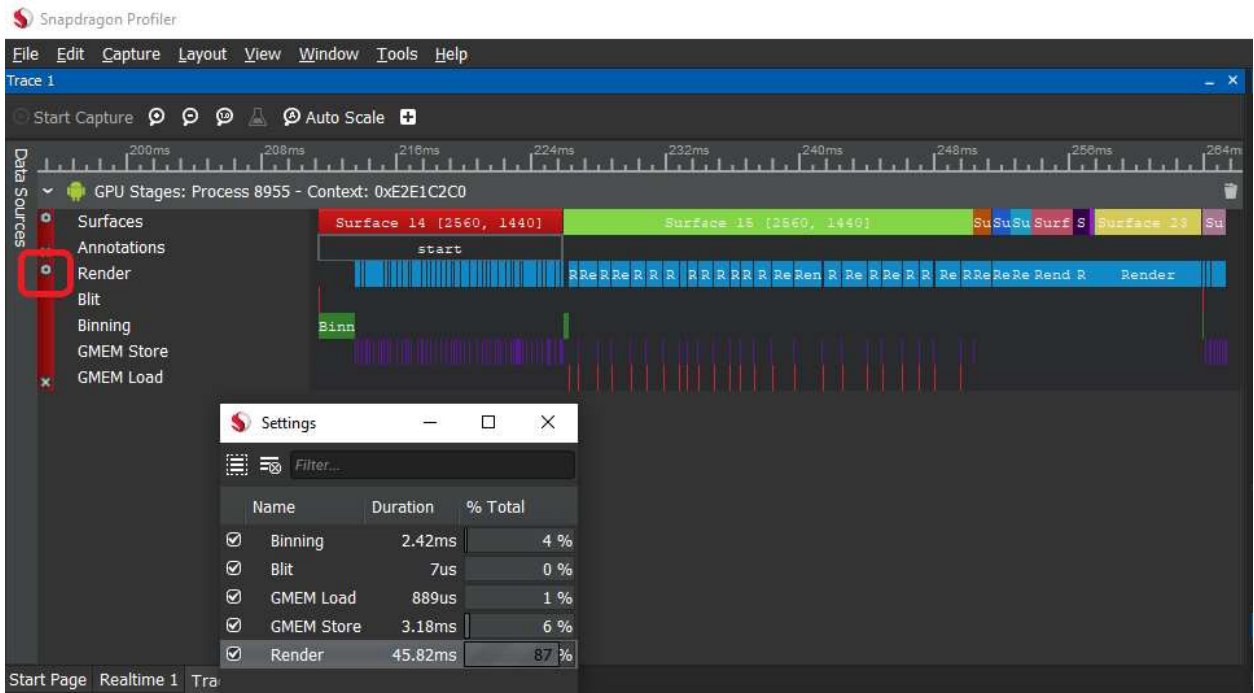


Figure 5-5 Measuring time with the calipers

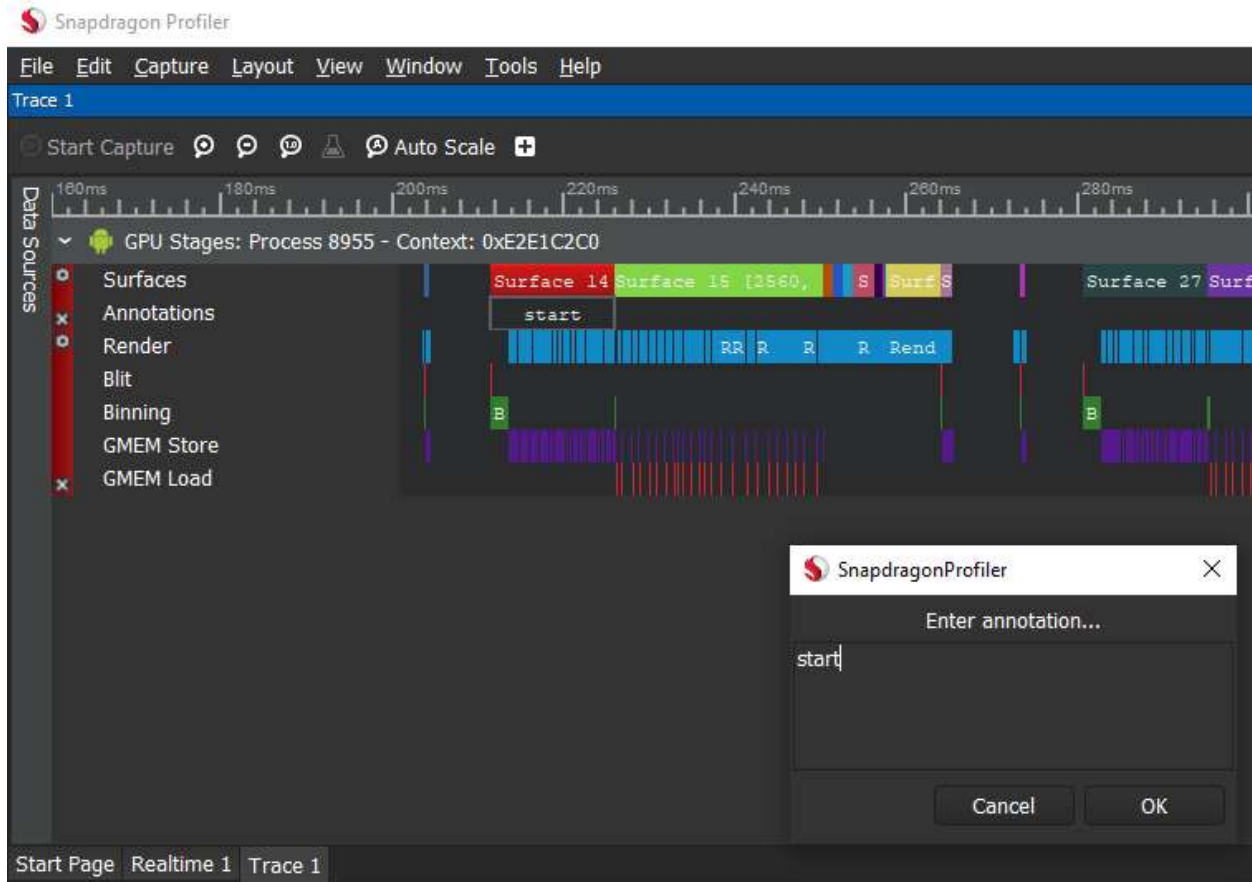
The **settings**, accessible via a small gear icon on the left side of the track, will summarize the amount of time spent in each track of that group. Only the elements visible on screen will be totaled.





## 5.2.9 Annotations

An annotation can be added to any block element in a trace capture by right clicking that element. An annotation track is added for that group.



## 5.3 Snapshot Capture

Snapshot Capture mode, shown in [Figure 5-6](#), allows the capture of a single frame of a graphics application. Snapshot Capture shows in detail how a scene is being rendered on the CPU.

In Snapshot Capture, choose a process to see the available metrics. Then select the metrics to view and take a snapshot. Once the frame is captured, view the captured data and step through the frame rendering draw call-by-draw call.

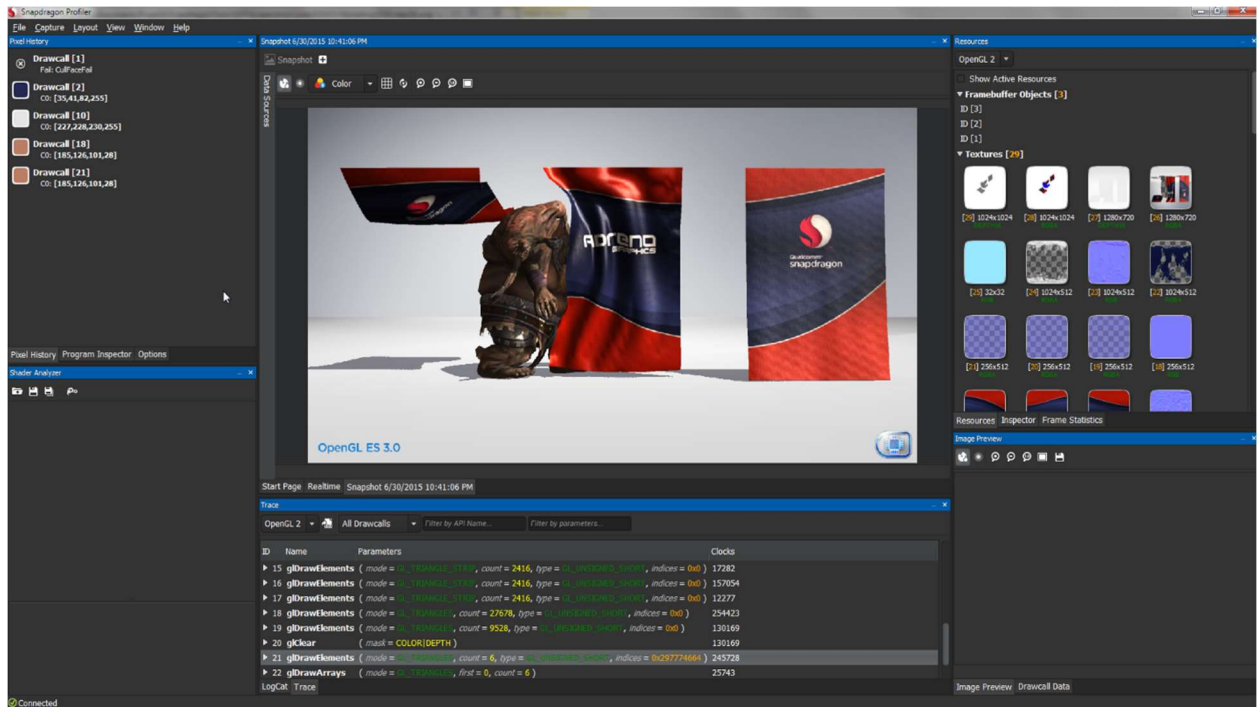


Figure 5-6 Snapshot Capture mode

### 5.3.1 Additional requirements

Both OpenGL ES and Vulkan apps must include *android.permission.INTERNET* and *android.permission.WRITE\_EXTERNAL\_STORAGE* in the app's AndroidManifest.xml.

If you wish to view SPIR-V assembly in the **Shader Analyzer** panel for Vulkan apps, install the **Android NDK** version 13b or above on the host machine and set its location in **File > Settings > Android > Android NDK Location**.

### 5.3.2 Launch applications

The **Launch Application** button in the **Snapshot Capture** panel can be used to launch any application installed on the target device. The implications of using this button depend on the type of app being launched.

- Vulkan apps *must* be launched using the button to enable capture of Vulkan snapshots.
- Using the button for OpenGL ES apps will allow Snapdragon Profiler to load the most current version of its OpenGL API interceptor library, which will enable optimal capture functionality. When OpenGL ES apps are launched on the device without using the button, the API interceptor library that is pre-installed on the device will be used. Depending on the age of that library it may lack functionality or contain bugs that have been fixed in the current version.
- All other apps behave the same whether they are launched from the device or with the button.

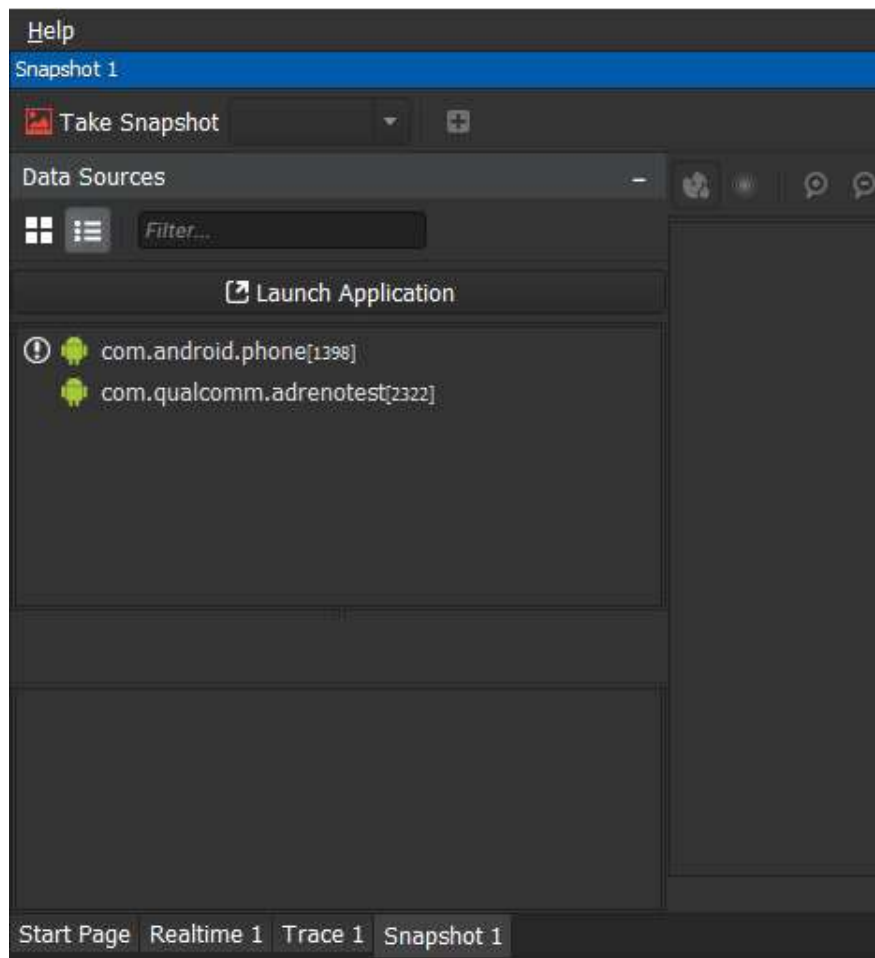
Launching an app using the **Launch Application** button makes use of the Java Debug Wire Protocol. Therefore, it is recommended to disable other debugging and profiling tools which use JDWP when using this functionality.

### 5.3.3 Snapshot Capture basics

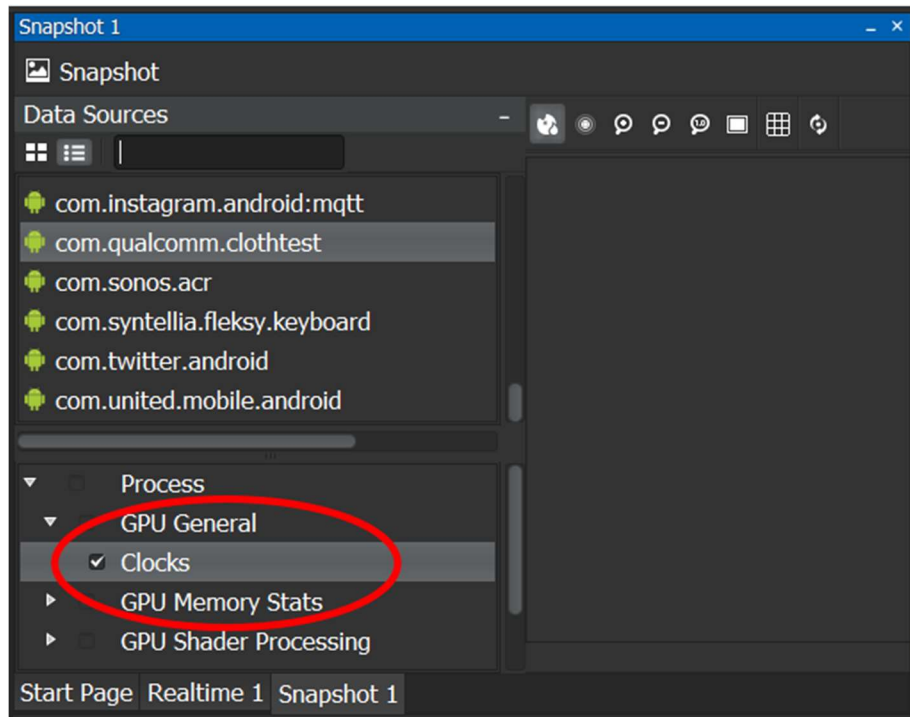
1. Check that the Android device is connected to the computer where Snapdragon Profiler is installed and launch the Profiler (see Section 3.2).

NOTE: Snapdragon Capture mode currently only works on mobile devices with a Qualcomm Snapdragon 805 (or later) processor and Android 6.0 (or later).

2. Choose **File > Connect**, or from the *Start Page*, click **Connect to a Device** to connect to a device to start a session (see Section 3.3).
3. From the *Start Page*, click **New Snapshot Capture**, or choose **Capture > New Snapshot**.
4. A new **Snapshot** tab is displayed:

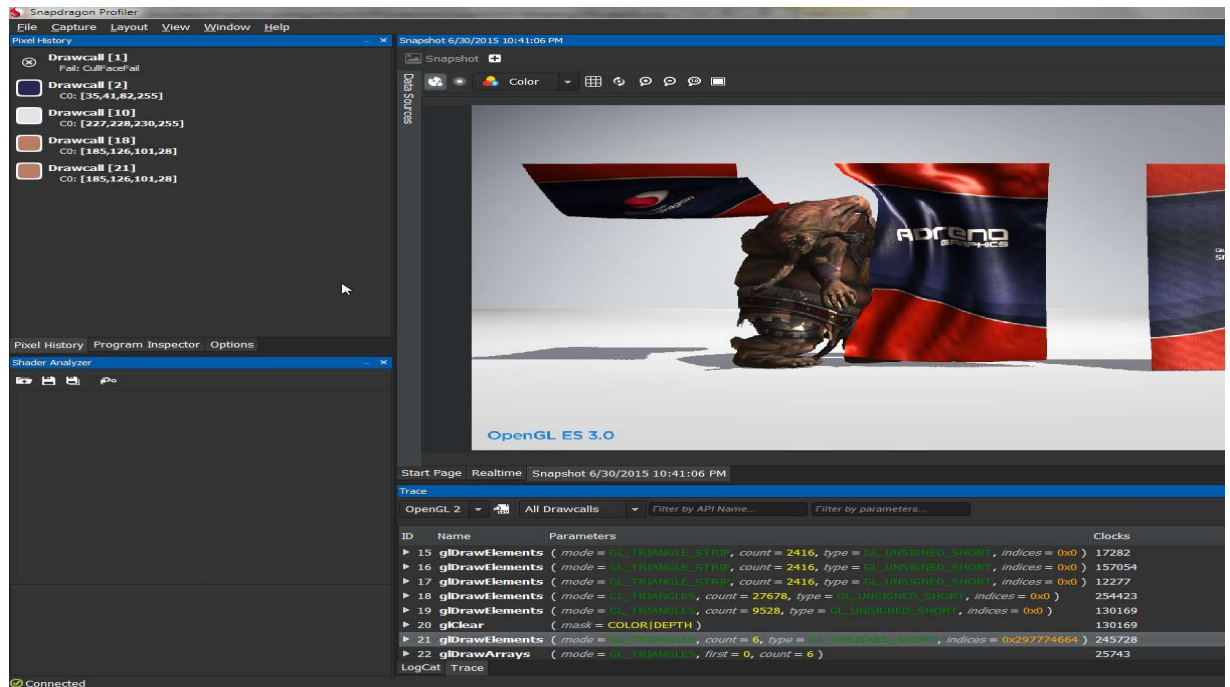


5. From the **Snapshot** tab, scroll through the **Data Sources** list and choose a process.
6. The metrics available for the process appear in the **Processes** list on the lower left.



NOTE: Vulkan Snapshot metrics for compute and secondary command buffers are not yet supported.

7. Click the metric to view in the snapshot, then click **Take Snapshot** to capture the frame.
8. An **orange bar** indicates the Profiler is retrieving the snapshot.
9. Once captured, view the captured data and step through the frame rendering drawcall-by-drawcall.

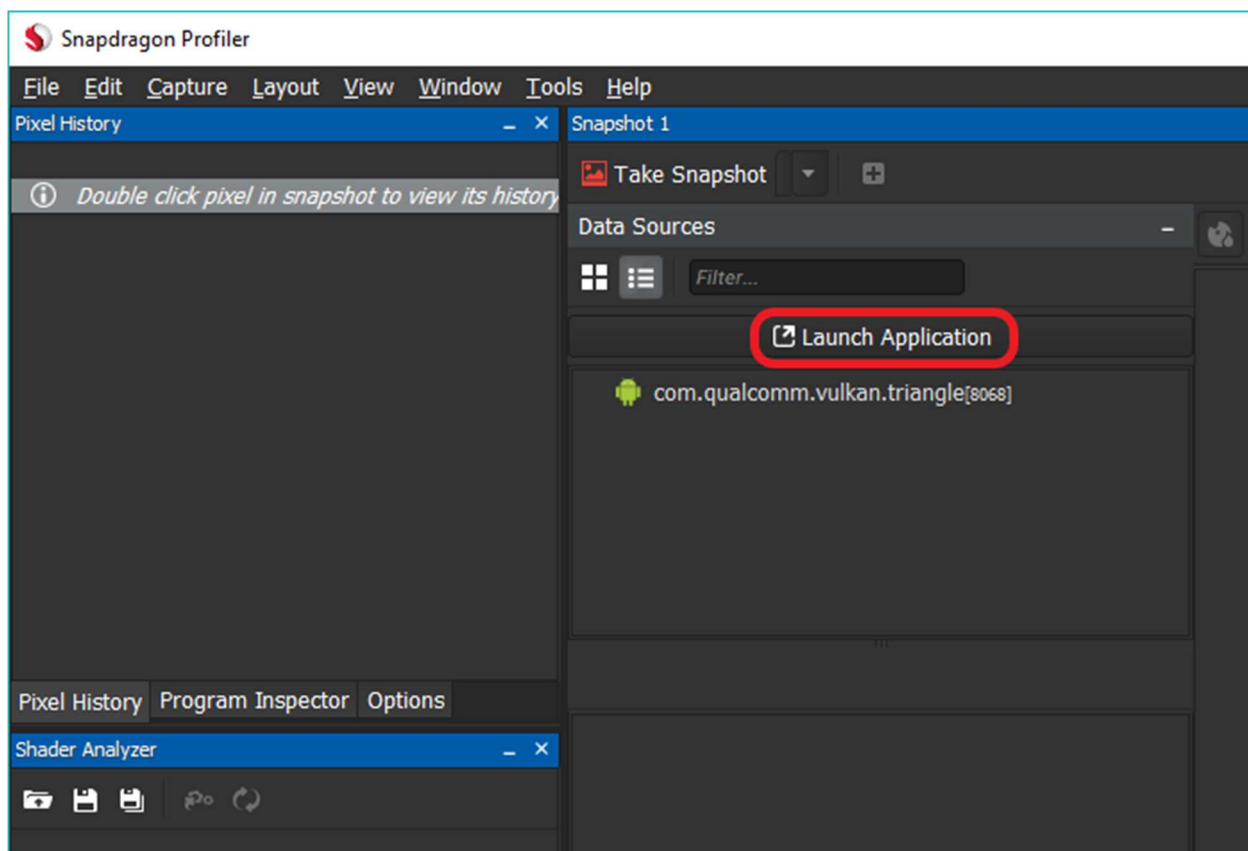


NOTE: Multiple snapshots can be taken in a session; however, **Pixel History**, **Drawcall Replay**, and **Frame Statistics** are only available for the most recent Snapshot Capture.

### 5.3.4 Vulkan Snapshot Capture

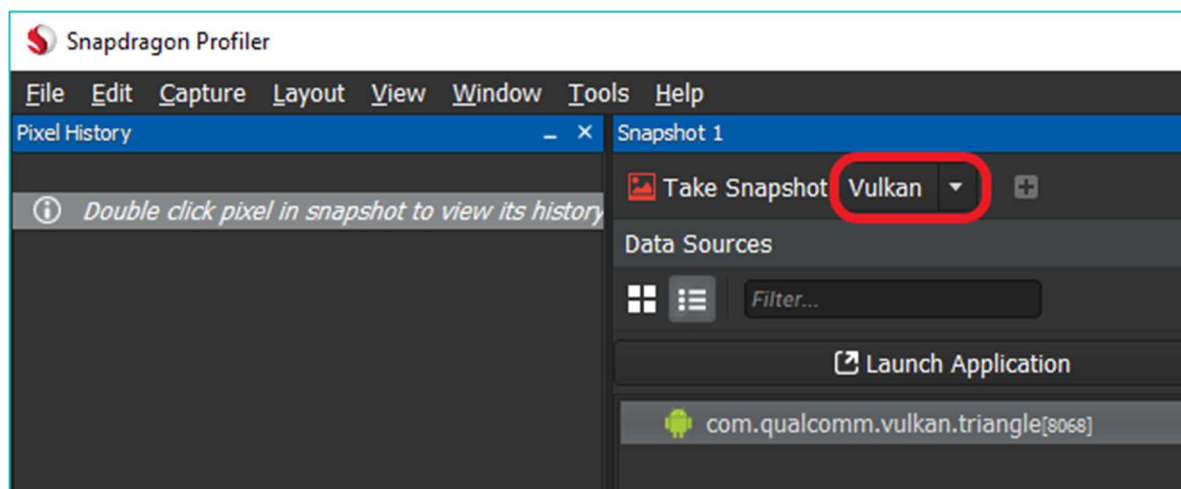
To capture a snapshot of a Vulkan application, the application *must* be launched from the **Snapshot Capture** panel of Snapdragon Profiler. Additionally, Vulkan Snapshot requires your app to be debuggable (except on rooted devices). Set the `android:debuggable` flag to true in the app's `AndroidManifest.xml`.

1. Follow steps 1-4 of [Snapshot Capture basics](#) to connect and open a new **Snapshot Capture** tab, then click the **Launch Application** button in the **Data Sources** panel.



2. The **Launch Application** window is shown in Figure 5-3. Select an application from the list of installed packages and click **Launch**. If the application is already running, it will restart with Vulkan snapshot profiling enabled.
3. Continue with step 5 of [Snapshot Capture basics](#) to select metrics and start a capture.

**NOTE:** If the selected application uses both OpenGL ES and Vulkan, select the graphics API you wish to capture in the combo box next to the **Take Snapshot** button.





## 5.4 Sampling Capture

Sampling Capture mode, shown in [Figure 5-7](#), captures a call graph for an application running on Snapdragon powered devices.

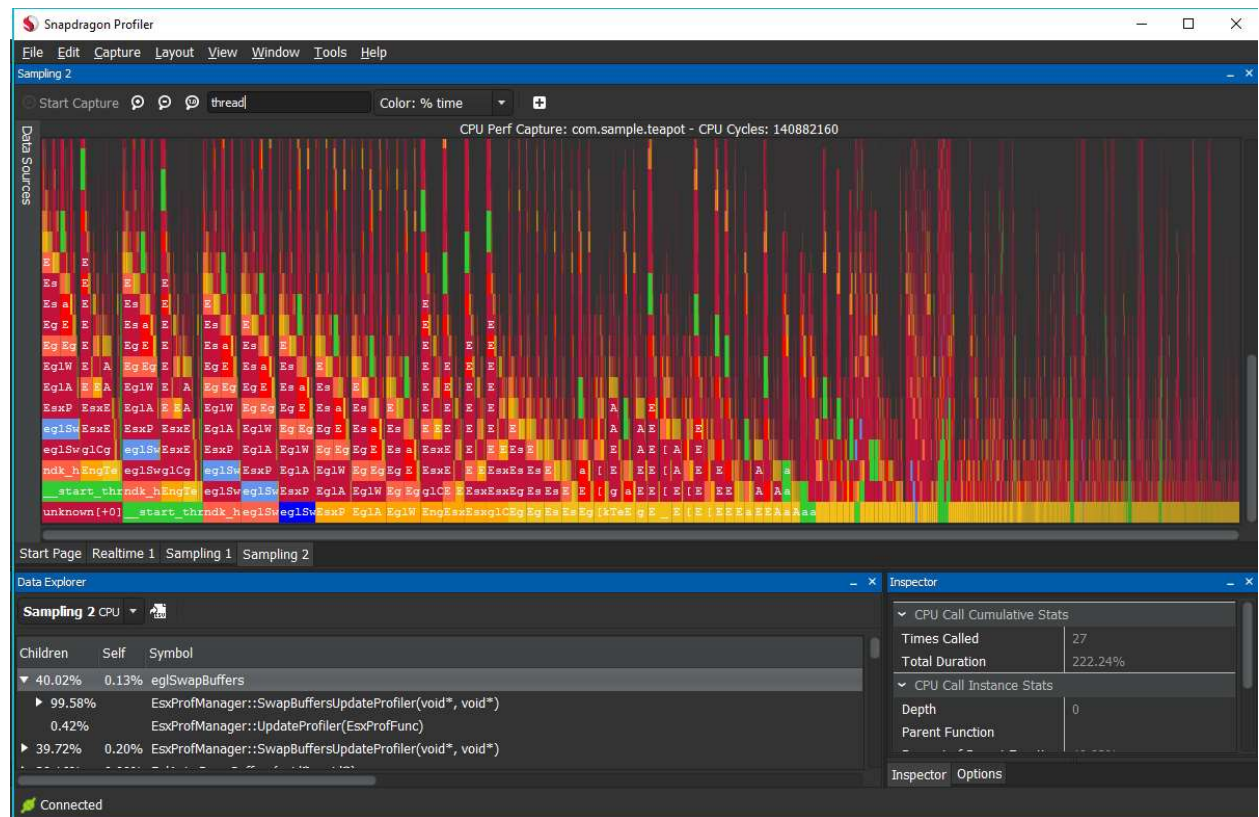
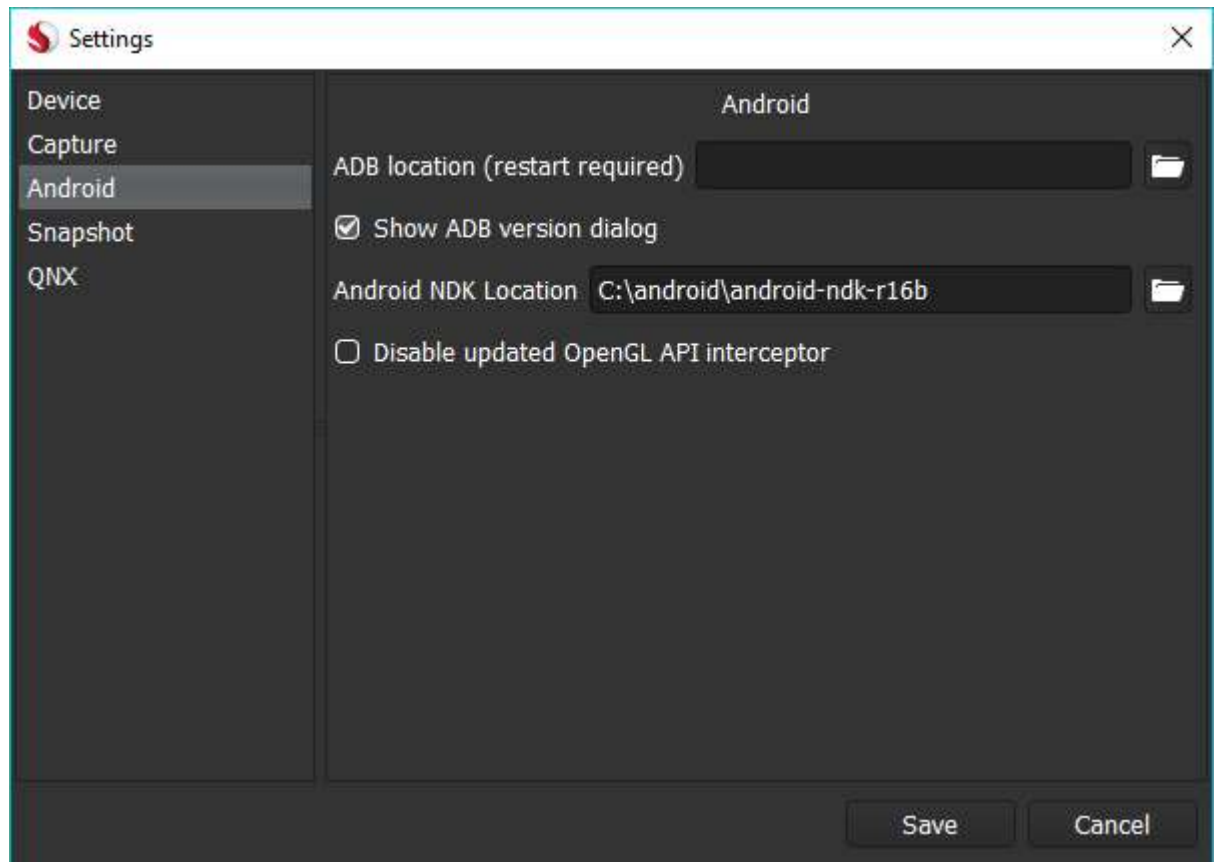


Figure 5-7 Sampling Capture mode

### 5.4.1 Additional requirements

Sampling Capture requires simpleperf, which is distributed through the **Android NDK** starting with revision r13b.

1. Install the **Android NDK** version 13b or above on the host machine.
2. Set the path to the **Android NDK** in the Snapdragon Profiler Settings (**File > Settings > Android > Android NDK Location**) and restart Snapdragon Profiler.



1. **Simpleperf** requires your app to be debuggable. Set the *android:debuggable* flag to true in the app's *AndroidManifest.xml*.

NOTE: Rooted Android devices do not require apps to have this flag.

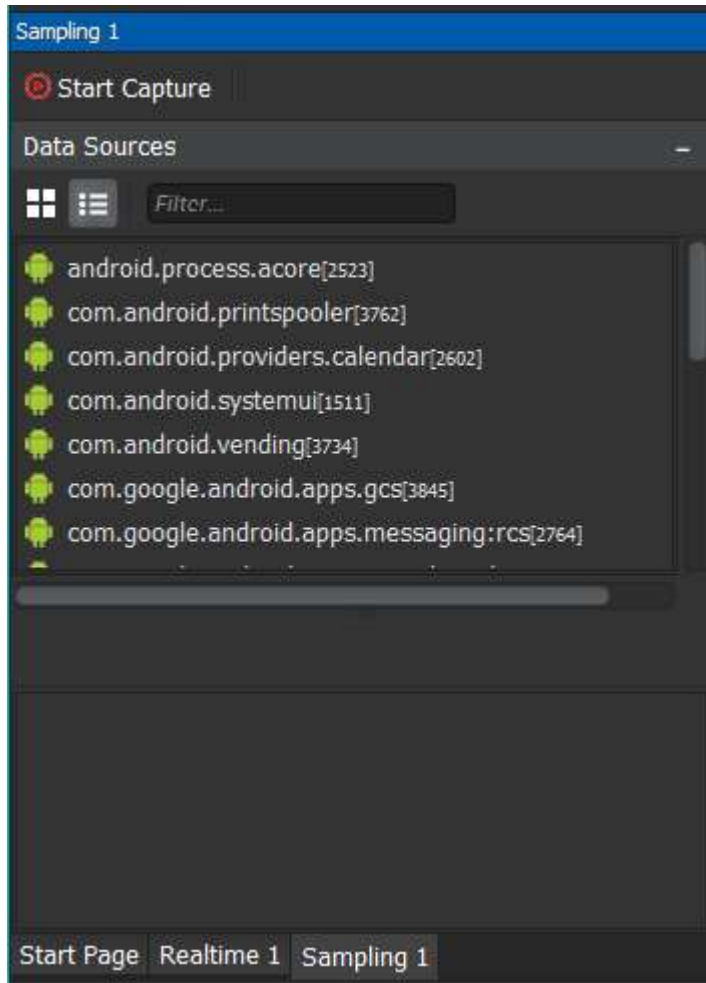
### 5.4.2 Sampling Capture basics

1. Check that the Android device is connected to the computer where Snapdragon Profiler is installed and launch the Profiler.

NOTE: Snapdragon Capture mode currently only works on Android 7.0 (or later)

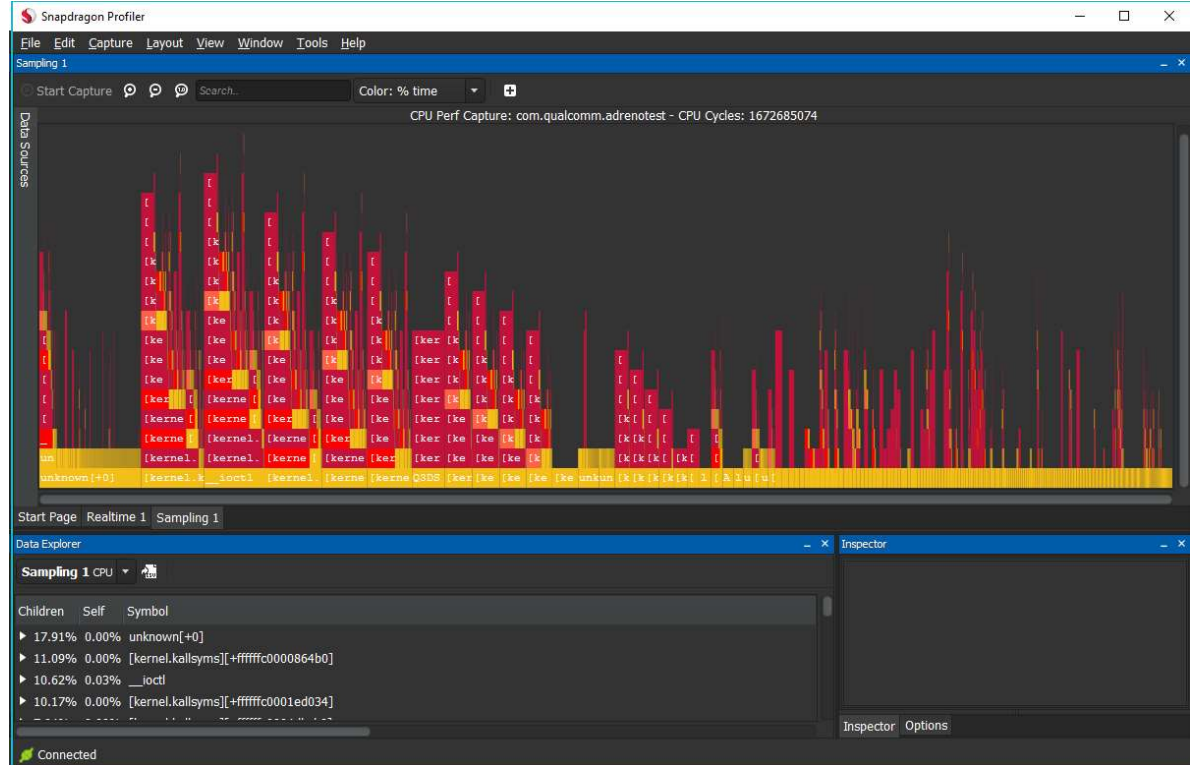
2. Choose **File > Connect**, or from the *Start Page*, click **Connect to a Device** to connect to a device to start a session.
3. From the *Start Page*, click **New Sampling Capture**, or choose **Capture > New Sampling**.
4. A new **Sampling Capture** tab is displayed:





5. From the **Data Sources** panel, choose a process from the **Process** list. Also choose a **Metric** specific to that process.
6. Click **Start Capture** to start the capture.

7. Click **Stop Capture** to stop the capture after a few seconds. Snapdragon Profiler processes and displays the call graph data in the main profiler window.



## 6 Saving Captures

---

All Capture types may be saved to disk.

1. Choose **File > Save**.
2. Enter a file name and select a location. Click the **Save** button. All captures in the session are saved to a file with the specified name. The default extension is *.sdp*.

To open a saved file:

1. Choose **File > Open**.
2. Select a Snapdragon Profiler capture file.
3. If the file contains a single capture, it will load automatically. Otherwise, a window will open, displaying a list of captures in the saved session. Select the capture you wish to load and click the **Open** button.
4. Repeat steps 1-3 to load additional captures in a Snapdragon Profiler capture file.

Snapdragon Profiler can open a session that was not explicitly saved.

1. Navigate to your <Documents>/SnapdragonProfiler folder.
2. Compress a <DateTime> folder containing a previous session in a .zip format.
3. Open Snapdragon Profiler and follow the instructions above to open a saved file.

**NOTE:** A device must be connected to open a snapshot capture. All other capture types may be opened without a device connection.

# 7 Troubleshooting

---

This chapter provides answers to common questions about using Snapdragon Profiler. If the issue is not addressed here, submit a question on the Qualcomm Developer Network, Snapdragon Profiler Support Forum at: <https://developer.qualcomm.com/forums/software/snapdragon-profiler>

## Why can't I connect to my device?

To resolve this issue:

1. Open a command prompt or terminal and confirm that the device is set up for ADB by running the **adb devices** command,
2. The device should display in the **List of devices attached** with a status of **device** (see Section 3.2).
3. Type **adb version** in the command prompt to confirm that ADB version 1.0.32 or later is installed.

Snapdragon Profiler relies on ADB to discover the connected devices. If ADB connection has been eliminated as the issue, contact us on the *Qualcomm Developer Network, Snapdragon Profiler Support Forum* at: <https://developer.qualcomm.com/forums/software/snapdragon-profiler>

## Why doesn't the track have any data when I choose a real-time metric?

If a metric is chosen that doesn't trigger system events to generate data, Snapdragon Profiler doesn't display any data.

## Why does nothing happen when I choose my application and run a Snapshot Capture?

Snapdragon Profiler interfaces with the graphics driver to query the information gathered in a Snapshot for a rendered frame. Some devices may not have the driver updates required to gather this data and properly support snapshots. Android 6.0 (or later) is required to use this feature.

## Why don't I see FPS on my device?

To view the FPS:

1. Select the process to see new categories in the Metrics list.
2. Expand the EGL category and FPS should be one of the metric options.

NOTE: FPS only works on OpenGL ES apps running on Snapdragon 805 (or later) mobile devices.

## Why don't I see Vulkan metrics?

To resolve this issue:

- Confirm the device is a supported device running Android N (or an Android 6.0 device with a graphics driver that supports Vulkan)
- Start the app using the **Launch Application** button in the **Trace Capture** panel or **Snapshot Capture** panel after connecting to Snapdragon Profiler.

## Why do I see unknown symbols when I run a Sampling Capture?

Make sure to build the app to include debuggable symbols. Also, kernel symbols are not available to user processes. More information is available in the Simpleperf Android Developers Guide at <https://developer.android.com/ndk/guides/simpleperf.html>, including how to profile Java code and apps build with Unity.

## I [do not see/am getting a warning about] my Vulkan application in the Data Sources window of Trace or Snapshot Capture.

- Vulkan Trace metrics and Vulkan Snapshot require your app to be debuggable. Set the *android:debuggable* flag to true in the app's AndroidManifest.xml. Rooted devices do not require apps to be debuggable. Both OpenGL ES and Vulkan apps must include *android.permission.INTERNET* in the app's AndroidManifest.xml.
- Snapdragon Profiler makes use of the Java Debug Wire Protocol to run an updated graphics shim on the device. We recommend disabling Android Studio and other debugging tools that use JDWP when starting an application from the **Launch Application** window.
- To take a Vulkan Snapshot, start (or restart) the app using the **Launch Application** button in the **Snapshot Capture** panel after connecting to Snapdragon Profiler. Apps launched from the Trace Capture panel must be restarted to take a snapshot.
- To take a Trace Capture with Vulkan metrics, start (or restart) the app using the **Launch Application** button in the **Trace Capture** panel after connecting to Snapdragon Profiler. Apps launched from the Snapshot Capture panel must be restarted to capture Vulkan Trace metrics.

**I see a performance hit running my app when I am no longer using Snapdragon Profiler on Android 10.**

Snapdragon Profiler may change settings on a device running Android 10 in order to profile your app. If the device is disconnected or rebooted before Snapdragon Profiler exits, the settings will not be deleted. For more information see

<https://developer.android.com/ndk/guides/graphics/validation-layer?release=r21#enable-layers-outside-app>

- Use adb to clear the settings SnapdragonProfiler may set during captures:

```
$ adb shell settings delete global enable_gpu_debug_layers
$ adb shell settings delete global gpu_debug_app
$ adb shell settings delete global gpu_debug_layers
$ adb shell settings delete global game_driver_prerelease_opt_in_apps
```

**I am able to take a Vulkan snapshot, but my app crashes when I select metrics with my snapshot, or when I perform a replay.**

Selecting metrics with your snapshot will automatically trigger a replay to record the metrics. If your app is memory intensive, the amount of memory required to keep both your app and Snapdragon Profiler alive during the replay may exceed the amount of memory available on some devices. You can check memory usage with 'adb shell top' or 'adb logcat -s lowmemorykiller'.