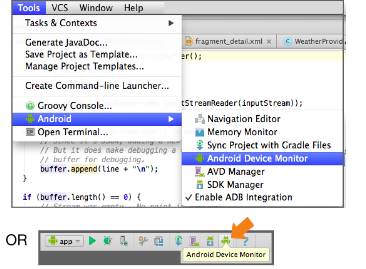
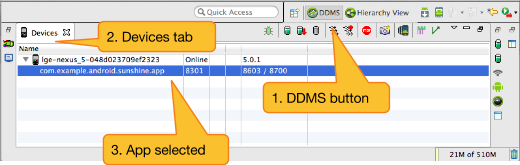
Working with Traceview



**Figure 1.**Steps for starting Android Device Monitor.

* 1. Connect your mobile device to your computer.
  2. Open your application in Android Studio, build the source, and run it on your device.
  3. Start the Android Device Monitor from Android Studio: **Tools -> Android -> Android Device** **Monitor**.

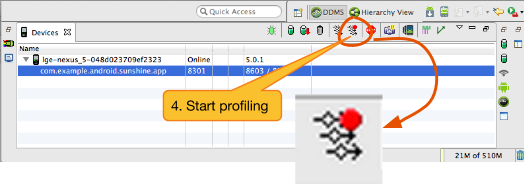


**Figure 2.**Steps for starting Traceview.

* 1. Make sure your device and the package for your application are showing in the **Devices** (DDMS mode) or **Windows** (Hierarchy Viewer mode) tab.

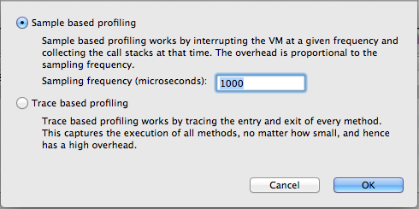
If necessary choose **Window > Reset Perspective** to get back to the default pane arrangement.

* 1. Click the **DDMS** button, because Traceview is one of the DDMS tools.
  2. Select the app you want to profile.



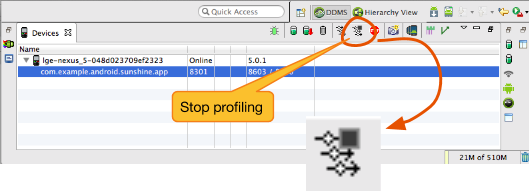
**Figure 3.**Starting profiling.

* 1. Click the Start method profiling https://developer.android.com/images/tools/performance/traceview/gettingstarted_image003.png button.



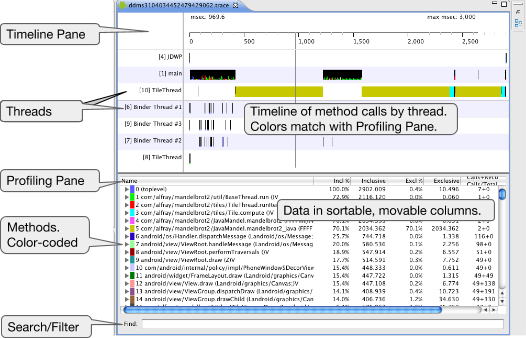
**Figure 4.**Choosing profiling options.

* 1. In the **Profiling Options** popup:
     + Choose **Sample based profiling**
     + Keep the default sampling rate of 1000 microseconds.
     + Click **OK**.



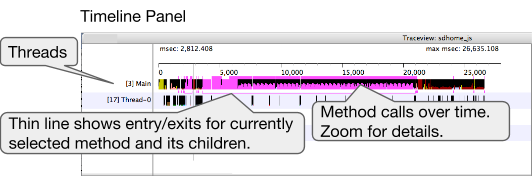
**Figure 5.**Stopping profiling.

* 1. Interact with your application. If you are aware of performance issues that your users experience, perform those actions.
  2. Click the Stop method profiling https://developer.android.com/images/tools/performance/traceview/gettingstarted_image006.png button.
  3. Wait for the trace to load. This may take a few seconds if you recorded for a long time.



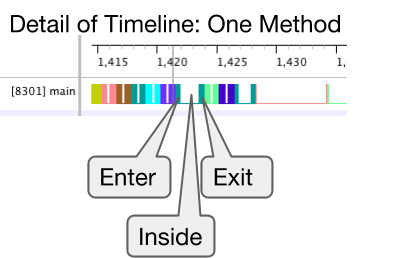
**Figure 6.**Traceview output panes.

* 1. Traceview has two panels: **Timeline** pane and **Profile** pane, which are described below.
  2. Use the **Find** box at the bottom to filter your profile results. For example, if you are interested in finding the running time for a function, you can search for it, and the tool will highlight it in the timeline.



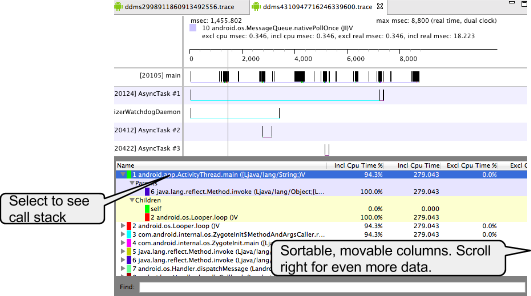
**Figure 7.**Annotated timeline panel.

* 1. The **Timeline** pane visualizes how your code executes over time.
     + Each row shows a thread.
     + Each bar on the timeline is a method executing.
     + Each color is for a different method; every time a method executes, you see a the same color bar.
     + The width of its bar indicates how long the method takes to execute.
  2. When you are zoomed out, a bar indicates when a method is executing.



**Figure 8.**Segment for method on the timeline.

* 1. Zoom into the graph, and the bar for each method expands into a colored U-shape, where the left side of the U indicates the start, and the right side the end of the method's execution.



**Figure 9.**Traceview Profiling pane with callstack.

* 1. The **Profiling** pane shows a list of methods.
     + Select a method to see who called it (Parent) and who it's calling (Children).
     + The selected method is also highlighted in the **Timeline** pane.
     + The columns show exclusive and inclusive CPU and real times, percentages, ratios, and how often a method was called.
     + The *exclusive time* is the time spent just in the method itself, which can help you find issues within that specific method.
     + The *inclusive time* is for the method and all methods it calls, which can help you find problems with your call tree.
     + The **Calls+Rec** column shows how many times a method was called recursively, which can help you track down performance issues.