PCMARK® for Android™

Technical Guide

Updated August 31, 2020



PCMark for Android **3**
Better benchmarking starts here 3
Using PCMark for Android 4
Minimum hardware requirements 7
How to benchmark performance 8
How to benchmark battery life 10
Work 2.0 benchmark **13**
Scoring 14
Web Browsing 2.0 15
Video Editing 16
Writing 2.0 18
Photo Editing 2.0 20
Data Manipulation 24
Work 2.0 battery life benchmark 26
Work benchmark **29**
Scoring 30
Web Browsing 31
Video Playback 32
Writing 34
Photo Editing 36
Work battery life benchmark 41
Computer Vision benchmark **44**
Scoring 45
TensorFlow 46
ZXing 47
Tesseract 48
Storage benchmark **50**
Implementation 51
Workload tasks 52
Scoring 53
Reporting scores **56**
Release notes **58**
About UL **61**

PCMARK FOR ANDROID

BETTER BENCHMARKING STARTS HERE

PCMark for Android is a benchmark for testing the performance and battery life of Android phones and tablets. PCMark measures performance using tests based on common, everyday tasks. See how well your device performs, then compare it with the latest models.

The big picture benchmark

- PCMark measures the performance of the device as a whole.
- PCMark tests are based on everyday activities, not abstract algorithms.
- See how well your device performs, then compare it with the latest models.

Results you can trust

- PCMark for Android is protected by public rules for manufacturers.
- We delist devices from our rankings that do not comply with our rules.
- All our benchmarks are backed by detailed technical guides.
- These guides explain what's being measured & how scores are calculated.

Make fair comparisons

- Our Best Devices list only includes devices that are available to the public.
- Each score is an average of all benchmarks results received for that model.
- We remove models that boost their scores with misleading optimizations.
- We never intentionally leak scores for prototype or pre-release hardware.

Created by experts

- UL has been creating industry standard benchmarks since 1997.
- PCMark is used throughout industry and by hundreds of publications.
- PCMark has been an industry standard PC benchmark since 2002.
- PCMark for Android was developed with leading technology companies.

USING PCMARK FOR ANDROID

PCMark for Android is a benchmarking app for smartphones, tablets and other Android devices.

In addition to producing benchmark scores, PCMark for Android helps you understand your results, track your score history, and compare your device with the latest models.

PCMark screen

The PCMark screen is the first page you see after opening the app. Start benchmarking your device by pressing one of the **RUN** buttons. Alternatively, swipe left or tap the button to see more tests on the Benchmarks screen.

Benchmarks screen

On this screen you can see which benchmarks are installed on your device. You can choose which benchmark tests you want to install.

Tap on a benchmark to open its Test Details screen.

Test Details

Each PCMark for Android benchmark contains a number of tests. The Test Details page explains which tests are included in the benchmark.

You can start a benchmark run by pressing the **RUN** button. Use the **UNINSTALL** button to free up storage space, while leaving the app installed and available to browse your score history and the Best Devices list.

My Device screen

This page shows your device's score history and system details.

- Highest scores from your own benchmark runs.
- History table showing past benchmark run dates and scores.
- Performance monitoring chart showing score history over time.
- Device details, features and specifications.

Score Details

Tap a score from the My Device Result History panel to open the Score Details view. This page displays the benchmark score, individual subtest scores, and some system details.

You can also view the hardware monitoring chart for the run. PCMark for Android polls the device once per second for CPU clock speed and

temperature and battery charge level while the benchmark is running. This monitoring has a negligible effect on performance.

Best Devices screen

The Best Devices screen is the ideal way to compare your device with the latest smartphones and tablets. By default, the page shows you a ranked list of devices ordered by performance.

Each score is an average of all the benchmarks results received for that model, with obvious outliers excluded. The average score for a device can be lower than the score from a clean benchmark run, as many people run benchmarks under less than ideal conditions. Since people use their devices in the real world and not a test lab, however, we believe that these averages best represent the true performance of each device.

You can re-order the list by name, performance, screen size, hardware (alphabetical), or popularity by tapping on the column headings.

The five star performance rating is calibrated so that the most powerful Android devices available score five stars. This means that a device's star rating may change over time as newer, more powerful devices are introduced.

Device popularity is based on the number of results submitted per model in the last 30 days.

Tap the **FILTERS** button to show more options. Use the **TEST** drop-down to see the list for a different test. Use the **ANDROID VERSION** drop-down to filter the list by a specific major version of Android. Use the **SEARCH** box to search the Best Devices list for a specific model, brand, CPU, GPU, SoC, or other text match. The search results include all devices that match your search plus your own device to enable easy comparison. For example, search for "nexus" to compare your device to all available Nexus devices.

Device details

Click on a device in the Best Devices list to open its Device Details page. This view shows your device and the selected device side-by-side for easy comparison.

Areas where a device is superior are highlighted green. Areas where a device is inferior are shaded red.

Help

The Help page includes an FAQ and a link to our support page.

Settings

Visit the Settings page to access the screen brightness calibration feature. You can also check the current version here and change the language.

MINIMUM HARDWARE REQUIREMENTS

OS	Android 5.0	
MEMORY	1 GB	
GRAPHICS	OpenGL ES 2.0 compatible	
DISPLAY	480 x 800 resolution	
STORAGE	PCMark app Work 2.0 benchmark Work benchmark Computer Vision benchmark Storage benchmark	75 MB 200 MB 400 MB 70 MB 100 MB

The requirements are not enforced by the app. It may be possible to run PCMark for Android on lesser hardware, however, it is not recommended or supported by UL.

HOW TO BENCHMARK PERFORMANCE

Before you start

In general, you should benchmark every device you test under the same conditions. For example, you should test every device in the same location, at room temperature, and away from direct sunlight and other heat sources.

The precision of UL benchmarks scores is usually better than 3%. This means that running a benchmark repeatedly on a consistently performing device in a well-controlled environment will produce scores that fall within a 3% range.

Individual scores may occasionally fall outside the margin of error since the factors that influence the score cannot be completely controlled in a modern, multitasking operating system. There are also devices that simply do not offer consistent performance due to their design. In these cases, it is necessary to run the benchmark multiple times, and then take either an average or a mode of the results.

To get accurate and consistent results you should close apps that may be running in the background, and disable notifications before running the benchmark. Some high-powered mobile devices use thermal throttling to avoid overheating the CPU, which can lead to lower scores on successive runs. To reduce this effect, we recommended waiting 15 minutes before and after PCMark runs to allow the device to cool down.

- Running other apps during the benchmark will affect the results.
- Avoid touching the screen while the tests are running.
- Press the Back Button if you want to stop the test.

Recommended process

1. Install all device updates to ensure your operating system is up to date.
2. Close other apps, especially those that run in the background.
3. Run the benchmark.

Expert process

1. Install all device updates to ensure your operating system is up to date.
2. Restart the device.
3. Wait 2 minutes for startup to complete.
4. Close all other apps, especially those that run in the background.
5. Wait for 15 minutes.

6. Run the benchmark.
7. Repeat from step 2 at least three times to verify your results.

HOW TO BENCHMARK BATTERY LIFE

Set the screen brightness

Screen brightness can have a significant effect on a device's battery life. To produce comparable results you should calibrate every device you test to the same screen brightness. In practice, this is difficult without expensive, specialist equipment.

Our benchmarks can display a pure white calibration screen to help you set the screen brightness. If you don't have access to a luminance meter, you can calibrate your devices by comparing the pure white calibration screen to a reference such as a monitor, light-box or similar. Alternatively, place your devices side by side and adjust the brightness levels by eye.

It is not a good idea to calibrate the screen using the device's built-in brightness settings. Different screens offer different levels of maximum brightness. The 50% brightness setting on one device may not be equal in luminance to the 50% setting on another device, for example.

Before you start

In general, you should benchmark every device you test under the same conditions. For example, you should test every device in the same location, at room temperature, and away from direct sunlight and other heat sources.

The battery must be at least 80% charged before the test will start. The test loops the benchmark until the battery charge drops below 20%. Battery testing can take several hours during which you will not be able to use your device for other tasks. Do not use the charging cable, or connect mobile devices to a PC, while the test is running.

To get accurate and consistent results you should close apps that may be running in the background, and disable notifications before running the benchmark.

- Running other apps during the benchmark will affect the results.
- Avoid touching the screen while the tests are running.
- Press the Back Button if you want to stop the test.

Recommended process

1. Test the device at room temperature, away from direct sunlight and heat sources.
2. Disable automatic screen brightness adjustment.

3. Use the pure white calibration screen to set the device's screen brightness to a defined level, ideally 200 cd/m2 for smartphones and tablets.

Expert process

To produce accurate, repeatable and comparable results you should test devices under controlled conditions. Our recommendations below match those given in the ECMA-383 standard "Measuring the Energy Consumption of Personal Computing Products."

1. Test the device in an environment that meets the requirements of ECMA-383:
    - Temperature: 23 +/- 5 degrees Celsius.
    - Relative humidity: 10 - 80%.
    - Ambient light: 250 +/- 50 lux.

2. Disable automatic screen brightness adjustment.

3. Use the pure white calibration screen in the app and a luminance meter to calibrate the screen brightness to 200 cd/m2 for smartphones and tablets.



WORK 2.0 BENCHMARK

See how your device handles common productivity tasks - browsing the web, editing videos, working with documents and data, and editing photos.

Use the **Work 2.0 performance benchmark** to test the performance of your device. This test takes around 10 minutes on a typical smartphone.

Use the **Work 2.0 battery life benchmark** to measure battery life and performance. This test can take many hours depending on the capacity and efficiency of the device.

Work 2.0 is an improved version of the older Work benchmark. It updates the Web Browsing, Writing, and Photo Editing tests and adds two all-new tests for Video Editing and Data Manipulation.

Web Browsing 2.0 measures performance when rendering a web page, scrolling, zooming, searching for content, and re-rendering the page after editing and adding an item. The test used the native Android WebView view.

Video Editing measures performance when playing, editing, and saving video. The test uses OpenGL ES 2.0, the native Android MediaCodec API, and Exoplayer, a Google-developed media player.

Writing 2.0 measures the time to open, edit, and save a document using the native EditText view and PdfDocument API. The PDF is encrypted, decrypted, and then rendered in a RecyclerView.

Photo Editing 2.0 measures the time taken to open, edit, and save a set of 4 MP JPEG images. The test uses four different APIs, including the latest version of the android.renderscript API, to filter and manipulate the images.

Data Manipulation simulates the demands of data-heavy applications such as fitness and financial apps. It measures the time to parse data from various file formats, then records frame rate while interacting with dynamic charts.

SCORING

The Work 2.0 performance benchmark reports a Work 2.0 performance score. Each sub-test also produces a score. Higher scores mean better performance.

$$Work\ 2.0\ score = geomean(WebBrowsing2.0, VideoEditing, Writing2.0, PhotoEditing2.0, DataManipulation)$$

Where:

$WebBrowsing2.0$	=	The result from the Web Browsing 2.0 test
$VideoEditing$	=	The result from the Video Editing test
$Writing2.0$	=	The result from the Writing 2.0 test
$PhotoEditing2.0$	=	The result from the Photo Editing 2.0 test
$DataManipulation$	=	The result from the Data Manipulation test

The Work 2.0 battery life benchmark reports battery life in hours and minutes and a Work 2.0 performance score.

⚠ Scores from Work 2.0 are not comparable with scores from the older Work benchmark test.

WEB BROWSING 2.0

The Web Browsing 2.0 test represents a common browsing scenario: checking up on friends on a social networking site. The test measures performance when rendering a web page, searching for content, and re-rendering the page after adding a new image.

Web Browsing 2.0 improves on the previous Web Browsing test by measuring frame rate as well as the time to complete the tasks.

Web Browsing 2.0 uses the native Android WebView view to display and interact with web pages. The content for these pages is stored in local files on the device. Android WebView uses the WebKit rendering engine. The test runs in portrait screen orientation.

A Web Browsing 2.0 score indicates how well the device performs when displaying and interacting with web content. It does not measure network speed, latency or bandwidth.

Workload tasks

1. Render the main page, which contains text and images of various sizes.
2. Scroll the page up and down.
3. Zoom into one picture.
4. Load a set of data into the search feature.
5. Execute searches as a search term is entered one character at a time.
6. Select a thumbnail area from a new image being added to the page.
7. Add the image to the page causing the page to be re-rendered.

Scoring

$$WebBrowsing2.0\ score = 2{,}500{,}000 \times\ geomean\left(\frac{1}{R_1}, \frac{1}{R_4}, \frac{1}{R_5}, \frac{1}{R_7}, A_2, A_4, A_7\right)$$

Where:

R_1	=	Time spent rendering the page for task 1
R_4	=	Time spent loading the data to be searched in task 4
R_5	=	Time spent searching the data in task 5
R_7	=	Time spent re-rendering the page in task 7
A_2	=	Average frame rate during task 2
A_4	=	Average frame rate during task 4
A_7	=	Average frame rate during task 7

VIDEO EDITING

Video Editing measures performance when playing, editing, and saving video. The test uses OpenGL ES 2.0, the native Android MediaCodec API, and Exoplayer, a Google-developed media player.

The first part of the test measures how well your device performs when applying real-time effects to video using OpenGL ES 2.0 fragment shaders.

The video frames are decoded and sent to an Android GLSurfaceView using a custom Renderer that applies OpenGL ES 2.0 shaders for the preview.

All video clips are encoded with the H.264/MPEG-4 AVC compression format. Several frame rates and resolutions are tested:

- 30 FPS at 1270 × 720
- 30 FPS at 1920 × 1080
- 30 FPS at 2560 × 1440
- 30 FPS at 3200 × 1800
- 60 FPS at 1270 × 720
- 60 FPS at 1920 × 1080
- 60 FPS at 2560 × 1440
- 60 FPS at 3200 × 1800

The second part of the test measures performance while decoding, editing, encoding, and muxing a video in the same code path. The video frames are decoded with a MediaCodec decoder on a SurfaceTexture using a custom Renderer that applies OpenGL shaders, then encoded with a MediaCodec encoder and muxed with MediaMuxer.

Scoring

$$VideoEditing\ score = 15{,}000\ \times geomean\left(geomean(R_1, R_2, R_3, R_4), \frac{1}{R_5}\right)$$

Where:

$$R_1 = \sum_{i=1}^{n} average_render_30FPS(i), (i = index\ of\ 30\ FPS\ video)$$

The sum of the average frame rate when rendering each 30 FPS video

$$R_2 = \sum_{i=1}^{n} average_decode_30FPS(i), (i = index\ of\ 30\ FPS\ video)$$

The sum of the average frame rate when decoding each 30 FPS video

$$R_3 = \sum_{i=1}^{n} average_render_60FPS(i), (i = index\ of\ 30\ FPS\ video)$$

The sum of the average frame rate when rendering each 60 FPS video

$$R_4 = \sum_{i=1}^{n} average_decode_60FPS(i), (i = index\ of\ 30\ FPS\ video)$$

The sum of the average frame rate when decoding each 60 FPS video

R_5 = Time taken to encode and mux the edited video

WRITING 2.0

The Writing 2.0 test measures how well your devices performs when working with documents. The test reflects the simple tasks you might perform on a mobile device such as adding text and images to a document as well as cutting, copying and pasting text within the document. It then asses the performance while creating and visualizing PDF documents.

The test uses the native Android EditText view, native PdfDocument and PdfRenderer Android API to create and then visualize a PDF file, and javax Cipher to encrypt and decrypt the file created. Document 1 is a 2.5 MB ZIP file containing a 100 KB text file and two 1.2 MB images. Document 2 is a 3.5 MB ZIP containing a 90 KB text file, a 1.6 MB image, and a 1.9 MB image.

Workload tasks

1. Load, uncompress and display Document 1.
2. Load, uncompress and display Document 2.
3. Copy all of Document 1 and paste it into the middle of Document 2. The combined document has around 200,000 characters.
4. Save Document 2 text and images to a ZIP file and compress. The file is about 6 MB.
5. Insert a 1.3 MB image to the middle of Document 2.
6. Save Document 2 text and images to a ZIP file and compress. The file is about 7.4 MB.
7. Save the document as a PDF in the external storage.
8. Encrypts the file with AES algorithm, CBC mode, PKCS#5 padding. Encryption key is generated with PBKDF2 variant with HmacSHA1.
9. Decrypts the file with the same algorithm and key.
10. Load the PDF file rendering the PDF pages on ImageViews inside a vertical-scrolling RecyclerView.

Scoring

$$Writing 2.0\ score = 2{,}700{,}000 \times geomean\left(\frac{1}{R_1}, \frac{1}{R_2}, \frac{1}{R_3}, \frac{1}{R_4}, \frac{1}{R_5}, \frac{1}{R_6}\right)$$

Where:

R_1 = The geometric mean of times measured to load Document 1 and Document 2.

R_2 = The sum of times measured for copying the text to the clipboard and pasting the text from the clipboard.

R_3 = The geometric mean of times measured to save Document 2 after the first copy paste operation and then in the last save in the end.

R_4 = The sum of times measured to add an image to the text.

R_5 = The geometric mean of the time to create the PDF document and to save it to a file with the time to load and render all the PDF pages.

R_6 = The geometric mean of the amount of time to encrypt and decrypt the file.

PHOTO EDITING 2.0

The Photo Editing 2.0 test measures how well your device performs when applying various filters and effects to images.

This test is the same as the one in the Work benchmark with one difference. Photo Editing 2.0 test uses the latest android.renderscript API in the place of the older android.support.v8.renderscript API that was used previously.

The Photo Editing 2.0 test uses four different APIs to apply filters and effects to the images.

1. The native android.media.effect API, which processes effects on the GPU.
2. The native android.renderscript API and specifically the RenderScript Intrinsics functions that support high-performance computation across heterogeneous processors.
3. The android-jhlabs API, which allows image processing using Java filters on the CPU, (licensed under the Apache license, version 2).
4. The native android.graphics API, which provides low-level tools that handle drawing to the screen directly.

The test measures the time taken to open, edit, and save a set of JPEG images. The measurements include the time taken to move graphics data to and from the CPU and GPU, decode from and encode to JPEG format, and read and write from the device's flash storage. The pictures used in the test have a 4 MP resolution (2048 × 2048).

Loading forces the use of bitmap format of `ARGB_8888`. Saving uses JPEG format with 90% quality hint. Saves happen to internal flash, unless `ENABLE_DUMP` configuration is set to permanently save all output pictures, in which case images are saved in the shared photo gallery area `DCIM/Futuremark/PCMA/PhotoEditing`.

Workload tasks

1. Load and display a source image.
2. Apply an effect to the source image and display the resulting image.
3. For selected images and filters, save the resulting image on disk.

Each of these tasks is followed by a one second delay to make it easier for you to follow the progress of the test.

Tasks 1 to 3 are repeated for 24 different filters, across 13 unique source images, with 6 intermediate saves.

In task 2, effects without parameters are run only once. Effects with parameters are run 3 times and the result is the arithmetic average of the execution times.

At the end of each Android media effect round (within task 2), `glFinish()` is called to make sure the results end up on the screen. This is also how applications work in order to keep user inputs and screen contents in sync.

Some of the parameterized effects are using each round to apply a different value (in the space (0,1]) to the effect, giving the impression of a 'phased' application. The values for the three rounds are: 0.33, 0.66, 1.0.

Effects, parameters and APIs

EFFECT NAME	PARAMETERS	API USED
autofix	phased	android.media.effects
blur	radius: 10.0	android.renderscript.ScriptIntrinsicBlur
contrast	phased	android.media.effects
contrast	phased	java
crop_java	x:0 y:0 w:1600 h:1600	android.graphics
crossprocess		android.media.effects
documentary		android.media.effects
emboss		android.renderscript.ScriptIntrinsicConvolve3x3
exposure	phased	java
fill light	phased	android.media.effects
fisheye	phased	android.media.effects
flip	horizontal	android.media.effects

EFFECT NAME	PARAMETERS	API USED
flip	horizontal	java
grain	phased	android.media.effects
grayscale		android.media.effects
lomoish		android.media.effects
rgb adjust	r:2 g:3 b:3	java
rotate	-90	android.media.effects
rotate_java	phased	android.graphics
saturate	phased	android.media.effects
sharpen		android.renderscript.ScriptIntrinsicConvolve3x3
sharpen	phased	android.media.effects
temperature	phased	android.media.effects
red eye correction		android.media.effects

Scoring

$$PhotoEditing2.0\ score = 800{,}000 \times geomean(R_j, R_r, R_a)$$

Where:

$$R_j = geomean\left(\frac{1}{ContrastJava}, \frac{1}{CropJava}, \frac{1}{FlipJava}, \frac{1}{RgbAdjustJava}, \frac{1}{RotateJava}\right)$$

$$R_r = geomean\left(\frac{1}{BlurRenderScript}, \frac{1}{SharpenRenderscript}, \frac{1}{EmbossRenderScript}\right)$$

$$R_a = geomean\left(\frac{1}{A_1}, \frac{1}{A_2}, \frac{1}{A_3}, \ldots, \frac{1}{A_{18}}, \frac{1}{A_{19}}, \frac{1}{A_{20}}\right)$$

And where:

A_1 = Time taken to apply Autofix effect

A_2 = Time taken to apply Contrast effect

A_3 = Time taken to apply Cross Process effect

A_4 = Time taken to apply Documentary effect

A_5 = Time taken to apply Fill-light effect

A_6 = Time taken to apply Fisheye effect

A_7 = Time taken to apply Flip effect

A_8 = Time taken to apply Grain effect

A_9 = Time taken to apply Greyscale effect

A_{10} = Time taken to apply Lomoish effect

A_{11} = Time taken to apply Rotate effect

A_{12} = Time taken to apply Saturate effect

A_{13} = Time taken to apply Sharpen effect

A_{14} = Time taken to apply Temperature effect

A_{15} = Time taken to apply Red Eye Correction filter effect

A_{16} = The time taken for face detection in the Red Eye Correction effect when scaling down the source image by a factor of four to 512 × 512 px and using the native Android FaceDetector API to find the location of the eyes. This is repeated three times with the arithmetic average stored as the result.

A_{17} = Arithmetic average of the loading time from flash to memory, including decompression for all images

A_{18} = Arithmetic average of the saving time from memory to flash, including image compression for all images

A_{19} = Arithmetic average of loading time from memory to GPU for all images

A_{20} = Arithmetic average of fetch times from GPU to memory for all saves

DATA MANIPULATION

The Data Manipulation test simulates the demands of data-heavy applications such as fitness and financial apps. It measures the time to parse data from various file formats, then records the frame rate while interacting with dynamic charts.

Workload tasks

The first part of the test focuses on measuring performance while reading, parsing, and validating data.

1. Read 10,000 tuples (date, value) of data from a CSV file with a standard InputStreamReader.
2. Read 10,000 tuples (date, value) of data from a XML file with Android-standard XmlPullParser.
3. Read 10,000 tuples (date, value) of data from a JSON file with common GSON library.
4. Read 10,000 tuples (date, value) of data from Protocol Buffers using Wire by Square Inc.

In the second part of the test, the data is visualized with dynamic, animated charts using the common open-source MPAndroidChart library. Several charts are tested with animations and common gestures such as swipe and zoom.

5. A cubic line chart is tested with three swipes forward and backwards, vertical and horizontal zoom-in/out and X/Y animations.
6. Bar chart is tested with three swipes forward and backwards, horizontal zoom-in/out and X/Y animations.
7. Pie chart is tested with swipe forward and backwards, circular swipe and circular animation.
8. Real-time cubic line chart is tested with a new data point added every 30 milliseconds.
9. Multiple line, bar, and cubic charts are shown inside a horizontal RecyclerView while performing scrolling animations.

Scoring

$$DataManipulation\ score = 5{,}000 \times geomean(P_1, P_2)$$

Where:

$$P_1 = geomean\left(\frac{1}{R_1}, \frac{1}{R_2}, \frac{1}{{R_3}^2}, \frac{1}{{R_4}^2}\right)$$

Where:

R_1 = Time to parse data from the CSV file in workload task 1

R_2 = Time to parse data from the XML file in workload task 2

R_3 = Time to parse data from the JSON file in workload task 3

R_4 = Time to parse data from the Protocol Buffer file in task 4

And where:

$$P_2 = geomean(C_1, C_2, C_3, C_4, C_5)$$

Where:

C_1 = The harmonic mean of the average draws/s for each part of workload task 5.

C_2 = The harmonic mean of the average draws/s for each part of workload task 6.

C_3 = The harmonic mean of the average draws/s for each part of workload task 7.

C_4 = The average draws/s for the duration of workload task 8.

C_5 = The harmonic mean of the average draws/s for each part of workload task 9.

WORK 2.0 BATTERY LIFE BENCHMARK

Measuring performance and battery life provides a better view of the overall profile of a device than benchmarking performance alone.

The Work 2.0 battery life benchmark uses the same workloads as the Work 2.0 performance benchmark.

The battery must be at least 80% charged before the test will start. The test loops the Work 2.0 performance benchmark until the battery charge drops below 20%. Do not use the charging cable or connect your device to a PC while the test is running.

Please refer to our best practice guide to set up your device and benchmark its battery life.

> ⚠ Battery life testing can take several hours during which you will not be able to use your device for other tasks.

Scoring

The reported battery life is an estimate for a 95% duty cycle (from 100% charged to 5%) extrapolated from the actual, measured battery life during the benchmark run:

$$Work\ Battery\ Life = 0.95 \times \frac{achieved_run_time}{battery_level_on_start\ -\ battery_level_on_end}$$

The Work 2.0 battery life benchmark also produces an overall Work 2.0 performance benchmark score, which is an average of the result from each loop while the battery test was running.

$$Work\ 2.0\ score = geomean(WebBrowsing2.0, VideoEditing, Writing2.0, PhotoEditing2.0, DataManipulation)$$

The subtest performance scores are calculated as geometric means of the results from test passes 1 to n:

$$WebBrowsing2.0 = geomean(Score_1, \ldots, Score_n)$$

$$VideoEditing = geomean(Score_1, \dots, Score_n)$$

$$Writing2.0 = geomean(Score_1, \dots, Score_n)$$

$$PhotoEditing2.0 = geomean(Score_1, \dots, Score_n)$$

$$DataManipulation = geomean(Score_1, \dots, Score_i)$$

The performance monitoring chart shows the performance score from each loop of the test. This makes it easy to see if performance decreases over time due to thermal issues or other factors.



WORK BENCHMARK

See how your device handles the work tasks you carry out every day - browsing the web, watching videos, working with documents, and editing photos.

Use the **Work performance benchmark** to test the performance of your device. This test takes around 20 minutes on a typical smartphone.

Use the **Work battery life benchmark** to test battery life and performance. This test can take many hours depending on the capacity and efficiency of the device.

Work includes four tests:

Web Browsing measures the time to render a web page, search for content, and re-render the page after editing and adding an item. The test used the native Android WebView view.

Video Playback measures the average frame rate during playback as well as the time to load, and seek within, 1080p video content using the native Android MediaPlayer API.

Writing measures the time to open, edit, cut, copy and paste text and images into a document using the native Android EditText view.

Photo Editing measures the time taken to open, edit, and save a set of 4 MP JPEG images while using four different APIs to filter and manipulate the images.

SCORING

The Work performance benchmark reports a Work performance score. Each sub-test also produces a score. Higher scores indicate better performance.

$$Work\ score = geomean(WebBrowsing, VideoPlayback, Writing, PhotoEditing)$$

The Work battery life benchmark reports battery life in hours and minutes and a Work performance score.

WEB BROWSING

The Web Browsing test represents a common browsing scenario: checking up on friends on a social networking site. The test measures the time it takes the device to render a web page, search for content, and re-render the page after adding a new image.

Web Browsing uses the native Android WebView view to display and interact with web pages. The content for these pages is stored in local files on the device. Android WebView uses the WebKit rendering engine. The test runs in portrait screen orientation.

A Web Browsing score indicates how well the device performs when displaying and interacting with web content. It does not measure network speed, latency or bandwidth.

Workload tasks

1. Render the main page, which contains text and images of various sizes.
2. Scroll the page up and down.
3. Zoom into one picture.
4. Load a set of data into the search feature.
5. Execute searches as a search term is entered one character at a time.
6. Select a thumbnail area from a new image being added to the page.
7. Add the image to the page causing the page to be re-rendered.

Scoring

$$WebBrowsing\ score = 2{,}500{,}000 \times\ geomean\left(\frac{1}{R_1}, \frac{1}{R_4}, \frac{1}{R_5}, \frac{1}{R_7}\right)$$

Where:

R_1	=	Time spent rendering the page for task 1
R_4	=	Time spent loading the data to be searched in task 4
R_5	=	Time spent searching the data in task 5
R_7	=	Time spent re-rendering the page in task 7

VIDEO PLAYBACK

The Video Playback test measures how well your device performs when asked to load, play, and seek within 1080p video content. The test uses the native Android MediaPlayer API and runs in landscape screen orientation.

The video content is 1080p resolution (1920 × 1080), encoded in H.264 using the baseline profile, level 4.0, with a variable bit rate target of 4.5 Mbps (max 5.4), at 30 FPS. The videos were encoded using Adobe Media Encoder CC using the Android - Tablet 1080p 29.97fps preset as the basis, but with the frame rate set to 30 FPS.

Workload tasks

1. Start playback of video 1
2. Play the video
3. Start playback of video 2
4. Play the video
5. Start playback of video 3
6. Play the video
7. Start playback of video 4
8. Seek within the video
    a. pause, seek 1000 ms forward, play, then repeat until 78000 ms is reached
    b. seek to 65000ms
    c. seek to 95000ms

Scoring

$$VideoPlayback\ score = 5{,}000 \times \left(\frac{1}{R_t} \times \left(\frac{1}{R_s}\right)^2 \times R_f^4\right)^{\frac{1}{7}}$$

Where:

R_t = The arithmetic average of the loading times measured in tasks 1, 3, 5 and 7. The measured time starts when the media framework is initializing and stops when the first frame is displayed.

R_s = The arithmetic average of seek times measured in task 8. The measured time starts when the seek operation is triggered and stops when the next video frame appears on the display.

R_f = The arithmetic average of average playback frame rates measured in tasks 2, 4 and 6.

WRITING

The Writing test measures how well your devices performs when working with documents. The test reflects the simple tasks you might perform on a mobile device such as adding text and images to a document as well as cutting, copying and pasting text within the document.

The test uses the native Android EditText view and runs in portrait orientation.

Document 1 is a 2.5 MB ZIP file containing a 100 KB text file and two 1.2 MB images. Document 2 is a 3.5 MB ZIP containing a 90 KB text file, a 1.6 MB image, and a 1.9 MB image.

Workload tasks

1. Load, uncompress and display Document 1.
2. Load, uncompress and display Document 2.
3. Copy all of Document 1 and paste it into the middle of Document 2. The combined document has around 200,000 characters.
4. Save Document 2 text and images to a ZIP file and compress. The file is about 6 MB.
5. Cut around 1,700 characters from the middle of Document 2 and paste to the beginning of the document. Cut a further 1,200 characters from the middle and paste near the beginning.
6. Type three short sentences near the start, middle and end of Document 2. The sentences range from 70 to 100 characters each.
7. Insert a 1.3 MB image to the middle of Document 2.
8. Save Document 2 text and images to a ZIP file and compress. The file is about 7.4 MB.

Scoring

$$Writing\ score = 2{,}700{,}000 \times geomean\left(\frac{1}{R_1}, \frac{1}{R_2}, \frac{1}{R_3}, \frac{1}{R_4}, \frac{1}{R_5}\right)$$

Where:

R_1 = The geometric mean of the loading times in tasks 1 and 2

R_2 = The geometric mean of the saving times in tasks 4 and 8

R_3 = Time taken to copy and paste text to and from the clipboard in task 3

R_4 = The sum of the times taken to copy text to the clipboard, cut it from the document, and paste into the document in task 5

R_5 = Time to insert an image into the document in task 7

PHOTO EDITING

The Photo Editing test measures how well your device performs when applying various filters and effects to images. The test runs in portrait screen orientation.

The test uses four different APIs to apply filters and effects to the images.

1. The native android.media.effect API, which processes effects on the GPU.
2. The native android.support.v8.renderscript API and specifically the RenderScript Intrinsics functions that support high-performance computation across heterogeneous processors.
3. The android-jhlabs API, which allows image processing using Java filters on the CPU, (licensed under the Apache license, version 2).
4. The native android.graphics API, which provides low-level tools that handle drawing to the screen directly.

The test measures the time taken to open, edit, and save a set of JPEG images. The measurements include the time taken to move graphics data to and from the CPU and GPU, decode from and encode to JPEG format, and read and write from the device's flash storage. The pictures used in the test have a 4 MP resolution (2048 × 2048).

Loading forces the use of bitmap format of `ARGB_8888`. Saving uses JPEG format with 90% quality hint. Saves happen to internal flash, unless `ENABLE_DUMP` configuration is set to permanently save all output pictures, in which case images are saved in the shared photo gallery area `DCIM/Futuremark/PCMA/PhotoEditing`.

Workload tasks

1. Load and display a source image.
2. Apply an effect to the source image and display the resulting image.
3. For selected images and filters, save the resulting image on disk.

Each of these tasks is followed by a one second delay to make it easier for you to follow the progress of the test.

Tasks 1 to 3 are repeated for 24 different filters, across 13 unique source images, with six intermediate saves.

In task 2, effects without parameters are run only once. Effects with parameters are run three times and the result is the arithmetic average of the execution times.

At the end of each Android media effect round (within task 2), `glFinish()` is called to make sure the results end up on the screen. This is also how applications work in order to keep user inputs and screen contents in sync.

Some of the parameterized effects are using each round to apply a different value (in the space (0,1]) to the effect, giving the impression of a 'phased' application. The values for the three rounds are: 0.33, 0.66, 1.0.

Effects, parameters and APIs

EFFECT NAME	PARAMETERS	API USED
autofix	phased	android.media.effects
blur	radius: 10.0	android.support.v8.renderscript.ScriptIntrinsicBlur
contrast	phased	android.media.effects
contrast	phased	java
crop_java	x:0 y:0 w:1600 h:1600	android.graphics
crossprocess		android.media.effects
documentary		android.media.effects
emboss		android.support.v8.renderscript.ScriptIntrinsicConvolve3x3
exposure	phased	java
fill light	phased	android.media.effects
fisheye	phased	android.media.effects
flip	horizontal	android.media.effects

EFFECT NAME	PARAMETERS	API USED
flip	horizontal	java
grain	phased	android.media.effects
grayscale		android.media.effects
lomoish		android.media.effects
rgb adjust	r:2 g:3 b:3	java
rotate	-90	android.media.effects
rotate_java	phased	android.graphics
saturate	phased	android.media.effects
sharpen		android.support.v8.renderscript.ScriptIntrinsicConvolve3x3
sharpen	phased	android.media.effects
temperature	phased	android.media.effects
red eye correction		android.media.effects

Scoring

$$PhotoEditing\ score = 800{,}000 \times geomean(R_j, R_r, R_a)$$

Where:

$$R_j = geomean\left(\frac{1}{ContrastJava}, \frac{1}{CropJava}, \frac{1}{FlipJava}, \frac{1}{RgbAdjustJava}, \frac{1}{RotateJava}\right)$$

$$R_r = geomean\left(\frac{1}{BlurRenderScript}, \frac{1}{SharpenRenderscript}, \frac{1}{EmbossRenderScript}\right)$$

$$R_a = geomean\left(\frac{1}{A_1}, \frac{1}{A_2}, \frac{1}{A_3}, \ldots, \frac{1}{A_{18}}, \frac{1}{A_{19}}, \frac{1}{A_{20}}\right)$$

And where:

A_1 = Time taken to apply Autofix effect
A_2 = Time taken to apply Contrast effect
A_3 = Time taken to apply Cross Process effect
A_4 = Time taken to apply Documentary effect
A_5 = Time taken to apply Fill-light effect
A_6 = Time taken to apply Fisheye effect
A_7 = Time taken to apply Flip effect
A_8 = Time taken to apply Grain effect
A_9 = Time taken to apply Greyscale effect
A_{10} = Time taken to apply Lomoish effect
A_{11} = Time taken to apply Rotate effect
A_{12} = Time taken to apply Saturate effect
A_{13} = Time taken to apply Sharpen effect
A_{14} = Time taken to apply Temperature effect
A_{15} = Time taken to apply Red Eye Correction filter effect

A_{16} = The time taken for face detection in the Red Eye Correction effect when scaling down the source image by a factor of four to 512 × 512 px and using the native Android FaceDetector API to find the location of the eyes. This is repeated three times with the arithmetic average stored as the result.

A_{17} = Arithmetic average of the loading time from flash to memory, including decompression for all images

A_{18} = Arithmetic average of the saving time from memory to flash, including image compression for all images

A_{19} = Arithmetic average of loading time from memory to GPU for all images

A_{20} = Arithmetic average of fetch times from GPU to memory for all saves

WORK BATTERY LIFE BENCHMARK

Measuring performance and battery life provides a better view of the overall profile of a device than benchmarking performance alone.

The Work battery life benchmark uses the same workloads as the Work performance benchmark.

The battery must be at least 80% charged before the test will start. The test loops the Work performance benchmark until the battery charge drops below 20%. Do not use the charging cable or connect your device to a PC while the test is running.

Please refer to our best practice guide to set up your device and benchmark its battery life.

> ⚠ Battery life testing can take several hours during which you will not be able to use your device for other tasks.

Scoring

The reported battery life is an estimate for a 95% duty cycle (from 100% charged to 5%) extrapolated from the actual, measured battery life during the benchmark run:

$$Work\ Battery\ Life = 0.95 \times \frac{achieved_run_time}{battery_level_on_start\ - battery_level_on_end}$$

The Work battery life benchmark also produces an overall Work performance benchmark score, which is an average of the result from each loop while the battery test was running.

$$Work\ score = geomean(WebBrowsing, VideoPlayback, Writing, PhotoEditing)$$

The subtest performance scores are calculated as geometric means of the results from test passes 1 to n:

$$WebBrowsing = geomean(Score_1, \ldots, Score_n)$$

$$VideoPlayback = geomean(Score_1, \dots, Score_n)$$

$$Writing = geomean(Score_1, \dots, Score_n)$$

$$PhotoEditing = geomean(Score_1, \dots, Score_n)$$

The performance monitoring chart shows the performance score from each loop of the test. This makes it easy to see if performance decreases over time due to thermal issues or other factors.



COMPUTER VISION BENCHMARK

Computer Vision is an exciting field that opens up the possibilities for a range of innovative apps and services. Recent advances mean that many mobile devices can now use these techniques. The Computer Vision test is a new test incorporating open-source software libraries that measures device performance for a range of image recognition tasks.

TensorFlow is an open-source machine learning library developed by Google. The test uses a trained neural network to recognize objects in a set of pictures.

ZXing, commonly known as Zebra Crossing, is a multi-format barcode image processing library. The test uses XZing to read a set of barcodes and QR codes.

Tesseract is an open-source optical character recognition library. The test recognizes and extracts English text from a set of images.

SCORING

The benchmark produces a Computer Vision score. Each sub-test also produces a score. Higher scores indicate better performance.

$$Computer\ Vision\ score = 1{,}000{,}000 \times geomean\left(\frac{1}{R_1}, \frac{1}{R_2}, \frac{1}{R_3}\right)$$

Where:

R_1	=	The result of the TensorFlow test.
R_2	=	The result of the ZXing test.
R_3	=	The result of the Tesseract test.

TENSORFLOW

This test uses a pre-trained neural network model to recognize objects in a set of photographic images. This technique has many practical uses in mobile applications such as identifying and classifying images in photography apps, tagging people and places in social networking apps, and helping visually impaired people understand the world around them.

A full explanation of machine learning, image recognition techniques, and neural networks is beyond the scope of this guide. We recommend following the links in the text below for further reading.

TensorFlow is an open-source machine learning library developed and supported by Google. "Inception" is a deep convolutional neural network architecture developed by Google for TensorFlow that is part of the current state of the art for computer image recognition.

The benchmark test uses a TensorFlow Inception model that has been pre-trained with the ImageNet database. ImageNet is an academic data set containing thousands of images that is commonly used for training image recognition systems.

The Inception model is about 52 MB in size, making it suitable for mobile devices. The model is loaded into memory and tested with a set of 10 images. Each image is reduced to 256 × 256 vector size before testing.

For each image, the model returns a percentage that represents its confidence that the image was recognized successfully. These percentages are not used in the scoring, however. Since the same model is used on all devices, the confidence of recognizing each image is also the same on all devices.

Instead, the test measures how long it takes the model to classify each image. The time taken depends on the performance of the processor in the device.

Scoring

The result is the average of the times taken to recognize each of the 10 images.

ZXING

ZXing, commonly known as Zebra Crossing, is an open-source, multi-format barcode image processing library.

This test uses XZing to read a set of four traditional barcodes as commonly found on product packaging and a series of seven QR codes. Each barcode and QR code has been photographed in poor lighting conditions or with simulated tearing damage to make the test a better representation of real-world use.

The test measures the time taken to recognize each barcode and QR code. It does not test the accuracy of the barcode image recognition.

Scoring

The result is the average of the times taken to recognize each barcode image.

TESSERACT

Optical character recognition (OCR) has many practical and useful applications from augmented reality translation apps like Google Translate to business card readers and document scanners.

Tesseract is an open-source optical character recognition library. This test uses a fork of Tesseract Tools for Android called tess-two.

This test use tess-two to recognize and extract English text from a set of four images. Each passage of text has been photographed in poor lighting conditions and at a slight angle to better simulate the challenges of OCR in real-world use.

The test measures the time taken to recognize and extract the text from each image. It does not test the accuracy of the OCR engine.

Scoring

The result is the average of the times taken to recognize the text in each image.



STORAGE BENCHMARK

A device's IO performance describes its ability to write data In to and read data Out of the storage. Good IO performance is key to a smooth, stutter-free experience. The PCMark for Android Storage benchmark is an isolating component benchmark for measuring the file IO and SQLite performance of the Android storage subsystem.

PCMark for Android measures storage IO performance in three areas:

Internal Storage is where your apps save private data such as settings and user data. The Android default cache directory is also in the internal storage. Files saved in internal storage are private to the application and cannot be accessed by the user or other applications. Internal storage performance most commonly impacts the start-up time and smooth running of your apps.

External Storage is used to save public data such as documents, photos, videos, and other files, as well as non-sensitive app data such as textures and sounds. Depending on the device, external storage can be removable, (such as an SD card), or built-in. Files in external storage can be found and modified by the user. External Storage performance most commonly impacts your experience when loading and viewing media files such as photos and videos.

The **Database** test measures performance when reading, updating, inserting and deleting database records using SQLite, the default relational database management system in Android. Following default Android behavior, the test database is saved in the device's internal storage.

IMPLEMENTATION

The Storage Test consists of two parts:

- File IO performance
- Android SQLite performance

File IO performance

In the File IO part, the test reads and writes files from and to the device's internal and external storage systems. The test performs both serial and random accesses with selected block sizes.

The data that is being read and written consists of random number characters. Before the read operations, the data is generated in the target partition by the application.

For each access pattern, the test is repeated 5 times, and the median bandwidth is used in the result calculation. The final result describes throughput measured in MB/s.

The implementation uses native code. The flags used when opening each file are `O_RDWR|O_CREAT|O_DIRECT`.

For the write operations, a call to `fsync()` follows each write.

Android SQL performance

In the Database part of the benchmark, the test performs a series of transactions for reading, updating, inserting, and deleting records in an SQLite database.

The reading part of the test reads 10 sets of 600 records, 6000 records in total. The update, insert, and delete parts use 2000 records. For each kind of operation, the bandwidth is used in the result calculation, given in IOPS (Input/Output Operations Per Second).

WORKLOAD TASKS

Internal storage workloads

1. Read a 32 MB file sequentially in 2 MB blocks from internal storage
2. Read a 32 MB file randomly in 4 KB blocks from internal storage
3. Write a 16 MB file sequentially in 2 MB blocks to internal storage
4. Write a 16 MB file randomly in 4 KB blocks to internal storage

External storage workloads

5. Read a 32 MB file sequentially in 2 MB blocks from external storage
6. Read a 32 MB file randomly in 4 KB blocks from external storage
7. Write a 16 MB file sequentially in 2 MB blocks to external storage
8. Write a 16 MB file randomly in 4 KB blocks to external storage

Database workloads

9. Read 10 sets of 600 records from an SQLite database table
10. Update 2000 records in an SQLite database table
11. Insert 2000 records into an SQLite database table
12. Delete 2000 records from an SQLite database table

SCORING

Overall Storage test score

The overall Storage benchmark score is calculated as follows:

$$Storage\ test\ score = 44 \times geomean(R_1, R_2 \times 2, R_3, R_4, R_5 \times 2, R_6, R_7, R_8, R_9 \times 2, R_{10} \times 2, R_{11} \times 2, R_{12} \times 2)$$

Where:

R_1	=	The arithmetic average of the read times from task 1
R_2	=	The arithmetic average of the read times from task 2
R_3	=	The arithmetic average of the write times from task 3
R_4	=	The arithmetic average of the write times from task 4
R_5	=	The arithmetic average of the read times from task 5
R_6	=	The arithmetic average of the read times from task 6
R_7	=	The arithmetic average of the write times from task 7
R_8	=	The arithmetic average of the write times from task 8
R_9	=	The IOPS measured in task 9
R_{10}	=	The IOPS measured in task 10
R_{11}	=	The IOPS measured in task 11
R_{12}	=	The IOPS measured in task 12

The weights used in the score formula are based on studies[1][2] that suggest that the most common I/Os issued by applications are:

- Random reads on the internal storage for loading application binaries, (represented by workload task 2).
- Sequential reads on the external storage for multimedia files playback, (represented by workload task 5).
- Database access for application internal data and Android contents, such as the dial log, contacts, web browsing logs, and multimedia databases, (represented by workload tasks 9, 10, 11, and 12).

[1] Lim. S. H., Lee S. & Ahn W. H. (2013) Applications IO profiling and analysis for smart devices. *Journal of Systems Architecture*, 59, 9, 740-747.

[2] Lee K. & Won Y. (2012) Smart layers and dumb result: IO characterization of an Android-based smartphone. *EMSOFT '12 Proceedings of the tenth ACM international conference on Embedded software*. ACM New York.

Secondary results

The table below shows how each secondary result related to a workload tasks, its unit, and its explanation.

TASK	RESULT	UNIT	EXPLANATION
1	Internal sequential read	MB/s	The median bandwidth over the results measured from the repeats of task 1
2	Internal random read	MB/s	The median bandwidth over the results measured from the repeats of the task 2
3	Internal sequential write	MB/s	The median bandwidth over the results measured from the repeats of the task 3
4	Internal random write	MB/s	The median bandwidth over the results measured from the repeats of the task 4
5	External sequential read	MB/s	The median bandwidth over the results measured from the repeats of the task 5
6	External random read	MB/s	The median bandwidth over the results measured from the repeats of the task 6
7	External sequential write	MB/s	The median bandwidth over the results measured from the repeats of the task 7
8	External random write	MB/s	The median bandwidth over the results measured from the repeats of the task 8

TASK	RESULT	UNIT	EXPLANATION
9	SQLite read	IOPS	The number of records read per second during task 9
10	SQLite update	IOPS	The number of record updates performed per second during task 10
11	SQLite insert	IOPS	The number of record insertions performed per second during task 11
12	SQLite delete	IOPS	The number of record deletions performed per second during task 12

REPORTING SCORES

Please follow these guidelines when including PCMark for Android scores in reviews or marketing materials to avoid confusing your customers and to ensure you represent our software correctly. Here's an example of how to do it right:

✓ "Smartphone scores 4,800 in PCMark for Android Work 2.0 benchmark."
× "Smartphone scores 4,800 PCMarks."

⚠ Scores from PCMark for Android are not comparable with other versions of PCMark.

Delisted Devices

We have clear rules for hardware manufacturers and software developers that specify how a platform can interact with our benchmark software. When a device is suspected of breaking those rules it is delisted. Scores from delisted devices are not shown in the Best Smartphones and Tablets list and should not be used to compare devices.

Using PCMark for Android scores in reviews

We provide complimentary Professional Edition benchmarks to members of the press working for established and reputable publications. Contact us at UL.BenchmarkPress@ul.com to request keys for your publication.

Press can use our benchmark scores in their hardware reviews. We kindly ask you to include a link to https://benchmarks.ul.com/ whenever you use our benchmarks in a review, feature or news story.

Using PCMark for Android scores in marketing material

For business purposes, a commercial license is granted with the purchase of PCMark for Android Professional Edition or a site license.

⚠ You must not disclose or publish PCMark for Android benchmark test results, nor may you use the UL logo or other UL assets in your sales and marketing materials, without prior, written permission from UL. Please contact UL.BenchmarkSales@ul.com for details.

On the first mention of PCMark for Android in marketing text, such as an advertisement or product brochure, please write "PCMark for Android benchmark" in order to protect our trademark. For example:

"We recommend the PCMark® for Android™ benchmark from UL."

Please include our legal text in your small print.

PCMark® is a registered trademark of Futuremark Corporation.

RELEASE NOTES

PCMark for Android v2.0.3716 – September 7, 2017

Compatibility

- Fixed an Android 8.0 compatibility issue in the Data Manipulation test. Benchmark scores are not affected.

PCMark for Android v2.0.3715 – June 27, 2017

Fixed

- Fixed an issue that could cause a crash when processing updates to the Best Devices list.
- Fixed a compatibility issue that caused a hang in the Video Editing test on Sony Xperia XA1.

PCMark for Android v2.0.3710 – January 30, 2016

Fixed

- Fixed an issue that could cause the app to hang before showing the results on very fast devices.

PCMark for Android v2.0.3706 – November 2, 2016

Fixed

- Fixed a rare issue that could cause the Photo Editing test in the Work 2.0 and Work benchmarks to crash.

PCMark for Android v2.0.3705 – October 13, 2016

New

- Benchmark your Android device with the new Work 2.0 benchmark, an improved version of the Work benchmark that updates the Web Browsing, Writing and Photo Editing tests and adds new Video Editing and Data Manipulation tests.
- Use the new Computer Vision benchmark to measure the performance of your device for a range of image recognition tasks.
- You can now filter the scores in the Best Devices list by major Android OS version number to see how performance is affected by software updates.

PCMark for Android v1.4.3539 - May 23, 2016

New

- Benchmark the storage performance of your Android device with the new Storage Test.

Fixed

- Restored the language option on the Settings screen. Choose from English, Chinese, and Russian.

Compatibility

- App now requires Android 5.0 or higher.

PCMark for Android v1.3.3083 - April 8, 2015

Fixed

- Fixed a bug in the Photo Editing test that could lead to a crash in extreme cases of low memory.

Compatibility

- New notification icon for improved appearance on Android 5.0 devices.

PCMark for Android v1.2.1781 - October 28, 2014

Improved

- See performance and battery life together in the Best Devices list.

Fixed

- Fixed an issue causing blank result screens on some devices.
- Fixed a bug that could show Simplified Chinese text on devices set to Traditional Chinese.

PCMark for Android v1.2.1773 - October 27, 2014

New

- Localized for Simplified Chinese.

Professional Edition only

- Added support for benchmark automation using adb (Android Debug Bridge) for Professional Edition license key holders.

PCMark for Android v1.1.1676 – October 2, 2014

Fixed

- Fixed an issue that could cause the Work Writing test to fail on some devices.

PCMark for Android v1.0.1646 – September 25, 2014

- Launch version.

Released on Google Play and as an APK for Benchmark Development Program partners and selected press publications.

ABOUT UL

UL is an independent, global company that offers a wide range of testing, inspection, auditing, and certification services. With 10,000 people in 40 countries, UL helps customers, purchasers, and policymakers navigate market risk and complexity. UL builds trust in the safety, security, and sustainability of products, organizations and supply chains – enabling smarter choices and better lives. Visit https://www.ul.com/ to find out more.

UL benchmarking software is developed by the Product Supply Chain Intelligence division. We enable global product compliance, innovation and promotion throughout the supply chain with our intelligent software and services backed by world-class scientific and technical expertise. Please visit https://psi.ul.com/ to find out more.

UL benchmarks help people measure, understand and manage computer hardware performance. Our talented team creates the industry's most trusted and widely used performance tests for desktop computers, notebooks, tablets, smartphones, and VR systems.

We work in cooperation with leading technology companies to develop industry-standard benchmarks that are relevant, accurate, and impartial. As a result, our benchmarks are widely used by the press. UL maintains the world's largest and most comprehensive hardware performance database, using the results submitted by millions of users to drive innovative online solutions designed to help people make informed purchasing decisions.

Our benchmarks are developed in Finland just outside the capital Helsinki. We also have a performance lab and sales office in Silicon Valley and sales representatives in Taiwan.

Press	UL.BenchmarkPress@ul.com
Sales	UL.BenchmarkSales@ul.com
Support	UL.BenchmarkSupport@ul.com

© 2020 Futuremark® Corporation. PCMark® trademarks, logos, character names, and distinctive likenesses are the exclusive property of Futuremark Corporation. UL and the UL logo are trademarks of UL LLC. Android is a trademark of Google Inc. All other names of companies and products mentioned herein may be trademarks of their respective owners.