Performance Testing Scripts for Android Oreo Go Edition - 3PL version

This toolkit allows partners to test and optimize Android Oreo Go Edition devices using Google's best practices.

The following areas are covered by the provided test scripts:

- Post boot metrics
- Kernel configurations
- Application startup time
- F2FS implementation

Google requires partners to submit the test results from these scripts in conjunction with CTS/GTS results before device approval is granted. The results can be submitted to your Google contact or/and 3PL contact.

1 Getting Started

1.1 Download `perf-scripts-<date>-<change>.tar.gz`

1.2 Extract `perf-scripts-<date>-<change>.tar.gz`

- Extract the package into a directory by using tar or any tool supporting tar/gunzip

Example:

```
google$ mkdir scripts
google$ cd scripts
google$ tar -xvzf <perf-scripts filename>
```

1.3 Device Setup

- Flash the device with the build you want to test
- Using either an existing account or creating a new google account
  - Login to the device with the google account
  - Sign in to contacts.google.com
  - Import file `perf_tests_contacts.csv` (included in the `docs/` directory in this package)
- Wait for contacts to sync before you proceed
- Turn off auto updates in Play Store
  - Navigate to Play Store app > Menu > Settings > General > Auto-update apps
  - Click on "Auto-update apps" and select "Do not auto-update apps"
- Turn off auto updates in settings
  - Settings > Users & Accounts
  - Turn off "Automatically Sync data"
- Allow enough time for device to finish setup on first boot before starting the tests. This usually will take from 2 minutes to 10 minutes depending on device and build.

2 Running the Scripts

2.1 `post_boot_metrics.sh` script (located in `system/`)

The `post_boot_metrics.sh` script measures several system metrics including free memory on boot. This script measures memory in the same way as the GTS test. If you have followed the setup instructions located in the GMS requirements then you should be able to pass the GTS test.

The script outputs the test results to the console and into the `post_boot_metrics-$timestamp.txt` file located in the `summary_results` directory.

Execute the script:

```
google$ cd system
google$ ./post_boot_metrics.sh
```

2.2 `verify_kernelcfg.sh` (located in `kernel/`)

The `verify_kernelcfg.sh` script validates the device kernel configuration against a known list of good/bad values. The kernel config values exists in `required_values.txt`.

Execute `verify_kernelcfg.sh` and record the output of the script.

Execute the script:

```
google$ cd kernel
google$ ./verify_kernelcfg.sh
```

2.3 `apps_performance.sh` (located in `apps/`)

Note: apps_performance.sh requires a device with Userdebug or rooted user build

The `app_performance.sh` script tests the startup performance of a number of applications.

This script requires all tests APKs to exist in the `apk/` directory. Create the directory under `app/` and download the APKs from here. If you do not have access, please request it via your Google/3PL contact.

2.3.1 Contents of the directory

- **Two .sh files** - `apps_performance.sh` and `test_app.sh`
- **Directory named `test_app_config`**, which must contain

  - a .config file for each apk we are going to test. The script runs for each APK that has a config file in the `test_app_config` directory
  - a file named `test_to_run.config`
  - All config files must be named in the format `$package_name.config`
- **Directory named apks**. This directory must contain
  - All the apks for which there is a config file in `test_app_config`
  - All apks must be named in the format `$package_name.apk`
  - APKs can be downloaded from here. If you do not have access, please request it via your Google/3PL contact.

Execute the script:

```
google$ cd apps
google$ ./apps_performance.sh
```

Note: You can run individual app tests by using `test_app.sh`.

2.4 `check_f2fs.sh` (located in `system/`)

This script/tool validates that the device is properly set up in f2fs and patches are correctly applied and work as expected.

Execute the script:

```
google$ cd system
google$ ./check_f2fs.sh
```

3 Sample Output

3.1 `post_boot_metrics.sh`

```
------------------------------------------------------------------------------

               RESULTS

------------------------------------------------------------------------------
Build fingerprint: google_gobo_gobo:8.1.0_OPM1.171019.011_03120951:userdebug_dev-keys
Device serial: A6R9K17624900506
MemAvailable: 523384 KB
MemAvailable + Dirty Cache: 611204 KB
Kernel Memory: 83280 KB
System server memory: 63268 KB
System UI Memory: 55808 KB
Launcher Memory: 41783 KB
Memory Carveout: 89660 KB
```

```
Total persistent Memory: 84000 KB

------------------------------------------------------------------------------

               Persistent Processes

------------------------------------------------------------------------------

55,808K:com.android.systemui
18,077K:com.android.phone
10,115K:****
13,492K:*********

Build approval metrics (Tested in GTS)
Device screen type: WVGA
Free memory test: PASS
Launcher memory test: OPTIMAL

Metrics impacting free memory and persistent memory
Kernel memory test: OPTIMAL
System server memory test: OPTIMAL
System UI memory test: OPTIMAL
Memory carveout test: OPTIMAL
Persistent memory test: PASS
```

3.2 Kernel Configs

- Lines starting with "Configuration missing" indicate that the configuration is missing
- Lines starting with "Unset or set to n" indicate that the config is set incorrectly and needs to be changed / flipped
- If you are using a chipset that does not support AES instructions, you can ignore the recommendation for "CONFIG_MTK_ADVERTISE_CE_SUPPORT"

```
Configuration missing: CONFIG_DM_VERITY=n
Configuration missing: CONFIG_HZ_300=y
Configuration missing: CONFIG_HZ=300
Unset or set to n: CONFIG_CGROUP_DEBUG
Unset or set to n: CONFIG_DYNAMIC_DEBUG
Unset or set to n: CONFIG_MMPROFILE
Unset or set to n: CONFIG_ZRAM_DEBUG
Unset or set to n: CONFIG_ZSM
```

3.3 App startup time

- Once your test is started, 2 additional directories will be created
  - `summary_results`
  - `test_results`
- `summary_results` directory contains a csv output for each test type run
- `test_results` directory contains a csv for each app and test type run
- The test types are defined in `test_to_run.config` under the folder `apps/test_app_config/`. Do not change `test_to_run.config` when submitting results to google.
- Please share both the `summary_results` and `test_results` directories with Google.

3.4 F2FS Tests

- If successful, the terminal will display a similar output to below
- The output will also be saved to a text file in `summary_results` under the directory where the f2fs scripts are stored
- The scripts provides a clear pass / fail output for mandatory features.
- Work with your Google contact if any of the tests fail. You may be missing some kernel F2FS patches in your build.

```
--------------------------------------------------------------------------------

                RESULTS

--------------------------------------------------------------------------------
Build fingerprint: google_gobo_gobo:8.1.0_OPM1.171019.011_03120951:userdebug_dev-keys
Device serial: A6R9K17624900506
F2FS support: PASS
F2FS status on /userata: PASS
Ipu policy: /sys/fs/f2fs/dm-2/ipu_policy
Discard granularity: /sys/fs/f2fs/dm-2/discard_granularity
Atomic write on /userata: PASS
Atomic write on /sdcard: PASS
Bad Write(2) Call: PASS
```