Qualcomm Technologies, Inc.

Perflock in Android O

80-NT384-2 C

October 17, 2017

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

Revision history

Revision	Date	Description
A	July 2017	Initial release
B	August 2017	Updated sections 2.1, 2.3, 2.4.1 and 3.4.1
C	October 2017	Updated Figure 2.1 and 2.2

Contents

Figures

Tables

1 Introduction

1.1 Purpose

This document describes the Boost Framework architecture feature and API implementation in Android O. The document provides a better understanding of Boost Framework and PerfHAL which are reworked because of hardware binder. This document explains the features along with the source code.

1.2 Conventions

Function declarations, function names, type declarations, attributes, and code samples appear in a different font, for example, `#include`.

Code variables appear in angle brackets, for example, `<number>`.

1.3 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://createpoint.qti.qualcomm.com/.

If you do not have access to the CDMATech Support website, register for access or send email to support.cdmatech@qti.qualcomm.com.

2 Overview

This chapter provides an overview of perflock mechanism.

2.1 Background

In Android O, Google introduces hardware binder so that OEMs can easily upgrade OS without updating the vendor libraries. To support this feature, QTI reworked the perflock implementation following Google’s changes.

In Android N, a socket interface was used for communicating between perfd client and server in the system partition. Therefore, if there are any changes or upgrades in Android Framework, then OEMs have to wait to get the update of perflock library from QTI for releasing their upgraded version of Android software. This is because perflock library is coupled with Android Framework source code.

Google separated system.img into two different images: vendor.img and system.img. System.img is the same. Vendor.img is newly introduced in Android O. It has vendor-specific libraries and files. QTI implemented new perfHAL in vendor.img to support this feature. Vendor.img is called vendor partition in this document.

2.2 Differences of perflock in Android N and Android O

- Modify Boost Framework
- Create PerfHAL instead of Perfd
- Move all perf resource files to vendor partition
- Create “libqti-perfd-client_system.so” shared library for using HIDL interface (hwbinder) in system partition
- Move `config.xml` to vendor partition and change name to `Perfboostsconfig.xml`
  - Android N
    - */device/qcom/msm8996/overlay/frameworks/base/core/res/res/values/**config.xml***
  - Android O
    - */vendor/qcom/proprietary/android-perf/configs/[chipset]/**perfboostsconfig.xml***

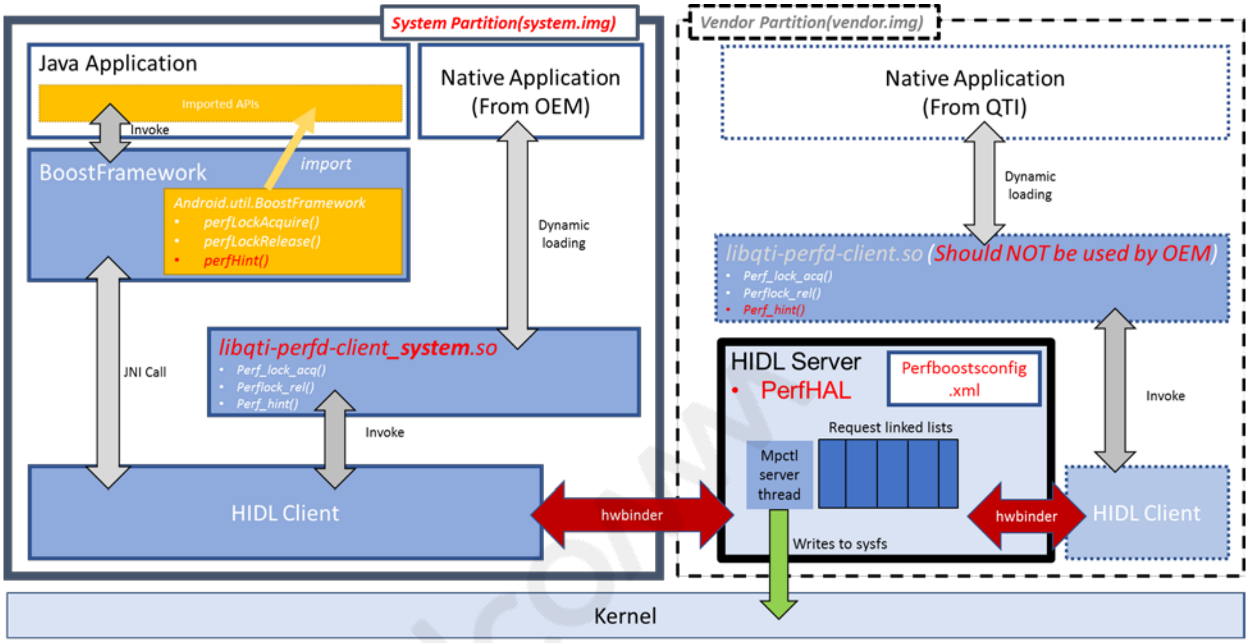

Figure 2-1 Android O - perflock architecture

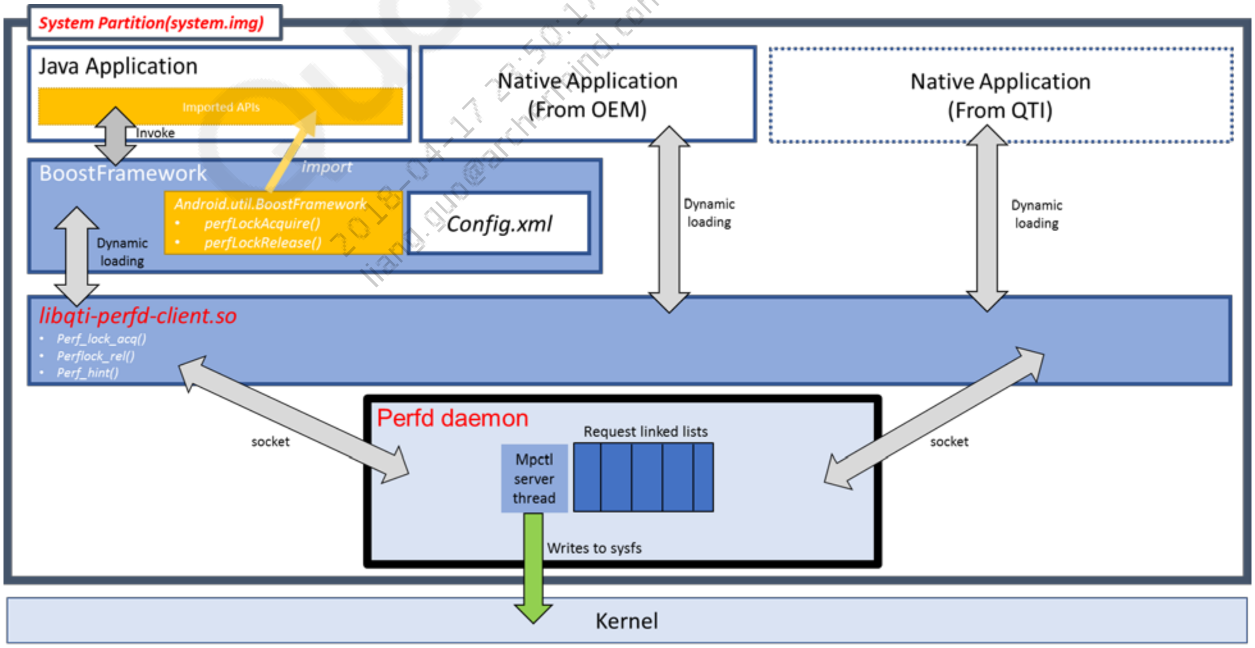

Figure 2-2 Android N - perflock architecture

2.3 Targets plan

Table 2-1 Targets Plan

Chipset	Android N	OTA(O) Nonbinderized HAL	New launch(O) Binderized HAL
SDM845	N/A	NO	YES
SDM660/SDM630	YES	NO	YES
MSM8998/8997	YES	NO	YES
MSM8996/8996Pro	YES	NO	YES
MSM8994/8992	YES	YES	NO
MSM8976/8976Pro/8956/8952	YES	YES	NO
MSM8953/8940/8937/8917/8917Cat6	YES	YES	NO
MSM8939/8916/8909	YES	YES	NO

2.4 Uses of Perflock

Perflock is used in the system partition in the following two ways:

2.4.1 Use boost framework for Java applications

1. Add import android.util.BoostFramework; in your Java source file
2. Create the BoostFramework class object
3. Use `perfLockAcquire` to request the optimizations required
4. Use `perfLockRelease` to release the lock
5. Use `perfHint` for hint-based request

2.4.2 Use native API for native applications

1. `libdl` and `libqti-perfd-client_system` has to be added to LOCAL_SHARED_LIBRARIES of the application's makefile.
2. Include `vendor/qcom/proprietary/android-perf/mp-ctl/mp-ctl.h` in LOCAL_C_INCLUDES.
3. Include dynamic linking header in source file, that is, `#include <dlfcn.h>`
4. Provide `libqti-perfd-client_system.so` library name for dlopen
5. Link the symbols `perf_lock_acq, perf_lock_rel` and `perf_hint` using dlsym
6. Declare an integer variable to store the handle returned by `perf_lock_acq` and `perf_hint`
7. Use `perf_lock_acq` to request the optimizations required
8. Use `perf_lock_rel` to release the lock
9. Use `perf_hint` to request hint trigger
10. Unload the library using dlclose on cleanup

3 Perflock API

This section explains the details of perflock interface.

3.1 Perflock acquire

Acquires the perflock with the list of optimizations requested.

3.1.1 Arguments

The new API requires the following arguments when perflock acquire is requested:

- handle, duration, <opcode value> pair
  - **Handle:** used to identify the client’s request.
    - Upon acquiring a perflock, the server acknowledges the request, and generate a unique integer to identify the request to return to the client
    - Upon releasing, the client passes in the handle returned by the server
    - A static variable has to be used to store the handle to ensure efficiency
  - **Duration:** dictates the maximum timeout period in milliseconds that the perflock should be held for.
    - Two types: definite and indefinite duration.
      - Definite locks require a positive integer value to specify the maximum timeout period. A timer is created and the perflock is released when the timer expires
      - Indefinite locks are held until the client calls the release function
    - If desired, perflock may be explicitly released before a timeout expires
  - **Opcode:** It is a 32-bit integer. Commonly used opcodes are detailed in the following table:

Table 3-1 Perflock resource and opcode

Perflock resource	Opcode
MPCTLV3_TOGGLE_POWER_COLLAPSE	0x40400000
MPCTLV3_MIN_FREQ_CLUSTER_BIG_CORE_0	0x40800000
MPCTLV3_MIN_FREQ_CLUSTER_BIG_CORE_1	0x40800010
MPCTLV3_MIN_FREQ_CLUSTER_BIG_CORE_2	0x40800020
MPCTLV3_MIN_FREQ_CLUSTER_BIG_CORE_3	0x40800030
MPCTLV3_MIN_FREQ_CLUSTER_LITTLE_CORE_0	0x40800100
MPCTLV3_MIN_FREQ_CLUSTER_LITTLE_CORE_1	0x40800110

Perflock resource	Opcode
MPCTLV3_MIN_FREQ_CLUSTER_LITTLE_CORE_2	0x40800120
MPCTLV3_MIN_FREQ_CLUSTER_LITTLE_CORE_3	0x40800130
MPCTLV3_MAX_FREQ_CLUSTER_BIG_CORE_0	0x40804000
MPCTLV3_MAX_FREQ_CLUSTER_BIG_CORE_1	0x40804010
MPCTLV3_MAX_FREQ_CLUSTER_BIG_CORE_2	0x40804020
MPCTLV3_MAX_FREQ_CLUSTER_BIG_CORE_3	0x40804030
MPCTLV3_MAX_FREQ_CLUSTER_LITTLE_CORE_0	0x40804100
MPCTLV3_MAX_FREQ_CLUSTER_LITTLE_CORE_1	0x40804110
MPCTLV3_MAX_FREQ_CLUSTER_LITTLE_CORE_2	0x40804120
MPCTLV3_MAX_FREQ_CLUSTER_LITTLE_CORE_3	0x40804130
MPCTLV3_SCHED_BOOST	0x40C00000
MPCTLV3_SCHED_PREFER_IDLE	0x40C04000
MPCTLV3_SCHED_MIGRATE_COST	0x40C08000
MPCTLV3_SCHED_SMALL_TASK	0x40C0C000
MPCTLV3_SCHED_MOSTLY_IDLE_LOAD	0x40C10000
MPCTLV3_SCHED_MOSTLY_IDLE_NR_RUN	0x40C14000
MPCTLV3_SCHED_INIT_TASK_LOAD	0x40C18000
MPCTLV3_SCHED_UPMIGRATE	0x40C1C000
MPCTLV3_SCHED_DOWNMIGRATE	0x40C20000
MPCTLV3_SCHED _MOSTLY_IDLE_FREQ	0x40C24000
MPCTLV3_SCHED_GROUP	0x40C28000
MPCTLV3_SCHED_SPILL_NR_RUN	0x40C2C000
MPCTLV3_SCHED_STATIC_CPU_PWR_COST	0x40C30000
MPCTLV3_SCHED_RESTRICT_CLUSTER_SPILL	0x40C34000
MPCTLV3_SCHED_FREQ_AGGR_GROUP	0x40C38000
MPCTLV3_SCHED_CPUSET_TOP_APP	0x40C3C000
MPCTLV3_SCHED_CPUSET_FOREGROUND	0x40C40000
MPCTLV3_SCHED_CPUSET_SYSTEM_BACKGROUND	0x40C44000
MPCTLV3_SCHED_CPUSET_BACKGROUND	0x40C48000
MPCTLV3_SCHED_SET_FREQ_AGGR	0x40C4C000
MPCTLV3_SCHED_ENABLE_THREAD_GROUPING	0x40C50000
MPCTLV3_SCHED_GROUP_UPMIGRATE	0x40C54000
MPCTLV3_SCHED_GROUP_DOWNMIGRATE	0x40C58000
MPCTLV3_SCHED_FREQ_AGGR_THRESHOLD	0x40C5C000
MPCTLV3_MIN_ONLINE_CPU_CLUSTER_BIG	0x41000000
MPCTLV3_MIN_ONLINE_CPU_CLUSTER_LITTLE	0x41000100
MPCTLV3_MAX_ONLINE_CPU_CLUSTER_BIG	0x41004000
MPCTLV3_MAX_ONLINE_CPU_CLUSTER_LITTLE	0x41004100
MPCTLV3_ABOVE_HISPEED_DELAY_INTERACTIVE_CLUSTER_BIG	0x41400000
MPCTLV3_BOOST_INTERACTIVE_CLUSTER_BIG	0x41404000
MPCTLV3_BOOSTPULSE_INTERACTIVE_CLUSTER_BIG	0x41408000
MPCTLV3_BOOSTPULSE_DURATION_INTERACTIVE_CLUSTER_BIG	0x4140C000

Perflock resource	Opcode
MPCTLV3_GO_HISPEED_DELAY_INTERACTIVE_CLUSTER_BIG	0x41410000
MPCTLV3_HISPEED_FREQ_INTERACTIVE_CLUSTER_BIG	0x41414000
MPCTLV3_IO_IS_BUSY_INTERACTIVE_CLUSTER_BIG	0x41418000
MPCTLV3_MIN_SAMPLE_TIME_INTERACTIVE_CLUSTER_BIG	0x4141C000
MPCTLV3_TARGET_LOADS_INTERACTIVE_CLUSTER_BIG	0x41420000
MPCTLV3_TIMER_RATE_INTERACTIVE_CLUSTER_BIG	0x41424000
MPCTLV3_TIMER_SLACK_INTERACTIVE_CLUSTER_BIG	0x41428000
MPCTLV3_MAX_FREQ_HYSTERESIS_INTERACTIVE_CLUSTER_BIG	0x4142C000
MPCTLV3_USE_SCHED_LOAD_INTERACTIVE_CLUSTER_BIG	0x41430000
MPCTLV3_USE_MIGRATION_NOTIF_CLUSTER_BIG	0x41434000
MPCTLV3_IGNORE_HISPEED_NOTIF_CLUSTER_BIG	0x41438000
MPCTLV3_ABOVE_HISPEED_DELAY_INTERACTIVE_CLUSTER_LITTLE	0x41400100
MPCTLV3_BOOST_INTERACTIVE_CLUSTER_LITTLE	0x41404100
MPCTLV3_BOOSTPULSE_INTERACTIVE_CLUSTER_LITTLE	0x41408100
MPCTLV3_BOOSTPULSE_DURATION_INTERACTIVE_CLUSTER_LITTLE	0x4140C100
MPCTLV3_GO_HISPEED_DELAY_INTERACTIVE_CLUSTER_LITTLE	0x41410100
MPCTLV3_HISPEED_FREQ_INTERACTIVE_CLUSTER_LITTLE	0x41414100
MPCTLV3_IO_IS_BUSY_INTERACTIVE_CLUSTER_LITTLE	0x41418100
MPCTLV3_MIN_SAMPLE_TIME_INTERACTIVE_CLUSTER_LITTLE	0x4141C100
MPCTLV3_TARGET_LOADS_INTERACTIVE_CLUSTER_LITTLE	0x41420100
MPCTLV3_TIMER_RATE_INTERACTIVE_CLUSTER_LITTLE	0x41424100
MPCTLV3_TIMER_SLACK_INTERACTIVE_CLUSTER_LITTLE	0x41428100
MPCTLV3_MAX_FREQ_HYSTERESIS_INTERACTIVE_CLUSTER_LITTLE	0x4142C100
MPCTLV3_USE_SCHED_LOAD_INTERACTIVE_CLUSTER_LITTLE	0x41430100
MPCTLV3_USE_MIGRATION_NOTIF_CLUSTER_LITTLE	0x41434100
MPCTLV3_IGNORE_HISPEED_NOTIF_CLUSTER_LITTLE	0x41438100
MPCTLV3_CPUBW_HWMON_MIN_FREQ	0x41800000
MPCTLV3_CPUBW_HWMON_DECAY_RATE	0x41804000
MPCTLV3_CPUBW_HWMON_IO_PERCENT	0x41808000
MPCTLV3_CPUBW_HWMON_HYST_OPT	0x4180C000
MPCTLV3_CPUBW_HWMON_LOW_POWER_CEIL_MBPS	0x41810000
MPCTLV3_CPUBW_HWMON_LOW_POWER_IO_PERCENT	0x41814000
MPCTLV3_CPUBW_HWMON_MAX_FREQ	0x41818000
MPCTLV3_CPUBW_HWMON_POLLING_INTERVAL	0x4181C000
MPCTLV3_CPUBW_HWMON_SAMPLE_MS	0x41820000
MPCTLV3_CPUBW_HWMON_IDLE_MBPS	0x41824000
MPCTL3_VIDEO_ENCODE_PB_HINT	0x41C00000
MPCTL3_VIDEO_DECODE_PB_HINT	0x41C04000
MPCTL3_VIDEO_DISPLAY_PB_HINT	0x41C08000
MPCTLV3_KSM_RUN_STATUS	0x42000000
MPCTLV3_KSM_PARAMS	0x42004000
MPCTLV3_SAMPLING_RATE_ONDEMAND	0x42400000

Perflock resource	Opcode
MPCTLV3_IO_IS_BUSY_ONDEMAND	0x42404000
MPCTLV3_SAMPLING_DOWN_FACTOR_ONDEMAND	0x42408000
MPCTLV3_SYNC_FREQ_ONDEMAND	0x4240C000
MPCTLV3_OPTIMAL_FREQ_ONDEMAND	0x42410000
MPCTLV3_ENABLE_STEP_UP_ONDEMAND	0x42414000
MPCTLV3_MAX_INTERMEDIATE_STEPS_ONDEMAND	0x42418000
MPCTLV3_NOTIFY_ON_MIGRATE	0x4241C000
MPCTLV3_GPU_POWER_LEVEL	0X42800000
MPCTLV3_GPU_MIN_POWER_LEVEL	0X42804000
MPCTLV3_GPU_MAX_POWER_LEVEL	0X42808000
MPCTLV3_GPU_MIN_FREQ	0X4280C000
MPCTLV3_GPU_MAX_FREQ	0X42810000
MPCTLV3_GPU_BUS_MIN_FREQ	0X42814000
MPCTLV3_IRQ_BALANCER	0X42C04000
MPCTLV3_UNSUPPORTED	0X42C00000
MPCTLV3_INPUT_BOOST_RESET	0x42C08000
MPCTLV3_SWAP_RATIO	0x42C0C000
MPCTLV3_STORAGE_CLK_SCALING_DISABLE	0x42C10000
MPCTLV3_KEEP_ALIVE	0x42C14000

  - Value: A new version opcode needs to be accompanied with a 32-bit value. Value should be in hexadecimal format, as currently perfd treat each value as hexadecimal.

3.1.2 Java API

3.1.2.1 Prototype

```
public int perfLockAcquire(int duration, int... list)
```

3.1.2.2 Parameters

- Duration
  - Duration can be definite or indefinite(0). If a duration is definite or timed, the perflock is released by the server after the duration is expired. If the duration is indefinite, user needs to call explicit perflockRelease for releasing the perflock.
- List
  - Resource opcodes and value pair.

3.1.2.3 return

This function returns 0 on success and -1 on failure.

3.1.3 Native API

Acquires the perflock with list of optimizations requested.

Enter all optimizations required into the list array; optimizations are applied in the ascending order of the array. For multiple distinct requests, use separate handles. The handle is used to track a distinct request. It must be passed in as an argument and also needs to be used to store the return value of perf_lock_acq.

3.1.3.1 Prototype

```
int perf_lock_acq(int handle, int duration, int list[], int numArgs)
```

3.1.3.2 Parameters

- Handle is used to track each distinct request, must be declared static.
- duration-time to hold lock in milliseconds, 0 for indefinite time.
  - Duration can be definite or indefinite(0). If a duration is definite or timed, the perflock is be released by the server after the expiration of the duration. If the duration is indefinite user needs to call explicit perf_lock_rel for releasing the perflock.
- list – an array of resource opcodes and value.
- numArgs – the number of elements in the list array.

3.1.3.3 Return

Returns: a nonzero integer as the handle on success, -1 on failure.

3.1.4 Reference

- If a resource is a frequency resource, value has to be specified in MHz. Example: If intended frequency is 1.5 GHz, the value to be requested is 1500, so specify 0x5DC.
- For a timer resource, value should be specified in milli second (msec).
  Example: 50 msec is the intended duration, the value to be requested is 50, so specify 0x32.
- For rest of the perflocks, the intended value should be specified.

Note: For some perflocks like MAX_CORE_ONLINE and INTERACTIVE_TIMER_RATE ONDEMAND_TIMER_RATE, the values are the absolute values. For example, if the requirement is to have only 1 core at max online, the value specified is 0xFE. With new perflock API the value specified is 0x1.

3.2 Perflock release

Releases the perflock held, which is identified with the handle that is passed as parameter. If the duration was set a call to perf_lock_rel is not mandatory.

3.2.1 Arguments

This API requires a handle to release the perflocks. Client has to pass the handle returned by perf lock acquire call to release complete request. The release value for all perflocks is 0xFFFFFFFF.

3.2.2 Java API

3.2.2.1 Prototype

```
public int perfLockRelease()
```

3.2.2.2 Parameters

No parameter

3.2.2.3 Return

On success, this function returns 0, and -1 on failure.

3.2.3 Native API

3.2.3.1 Prototype

```
int perf_lock_rel(int handle)
```

3.2.3.2 Parameters

Handle: used to track each distinct request. User pass in the same handle that was returned by perf_lock_acq to properly release the lock.

3.2.3.3 Return

This function returns 0 on success and -1 on failure.

3.3 Perf Hint

XML lookup is performed based on hint ID, and hint type, and a perflock call is requested with the specified parameter in xml file.

3.3.1 Arguments

The following is the new API introduced in Android O. It requires the following arguments:

- "hint_id, package_name, duration, hint_type/reserved"
  - □ `Hint`: Id identifying each action uniquely, like launch, scroll.
  - □ `Package Name`: Package name of the app, who triggered the hint
  - □ `Duration`: Duration of the perflocks mapped to that hint.
  - □ `Reserved(Hint Type):` To differentiate between similar action, like horizontal scroll vs. vertical scroll
- The list of hints: Currently these are the defined hints.

Table 3-2 PerfHints

VENDOR_HINT_START = 0x00001000,
//activity trigger hints
VENDOR_ACT_TRIGGER_HINT_BEGIN = 0x00001001,
VENDOR_HINT_ACTIVITY_START = 0x00001001,
VENDOR_HINT_ACTIVITY_STOP = 0x00001002,
VENDOR_HINT_ACTIVITY_RESUME = 0x00001003,
VENDOR_HINT_ACTIVITY_PAUSE = 0x00001004,
VENDOR_HINT_PROCESS_START = 0x00001005,
VENDOR_HINT_NET_OPS = 0x00001006,
VENDOR_ACT_TRIGGER_HINT_END = 0x0000107F,
//perf hints
VENDOR_PERF_HINT_BEGIN = 0x00001080,
VENDOR_HINT_SCROLL_BOOST = 0x00001080,
VENDOR_HINT_FIRST_LAUNCH_BOOST = 0x00001081,
VENDOR_HINT_SUBSEQ_LAUNCH_BOOST = 0x00001082,
VENDOR_HINT_ANIM_BOOST = 0x00001083,
VENDOR_HINT_ACTIVITY_BOOST = 0x00001084,
VENDOR_HINT_TOUCH_BOOST = 0x00001085,
VENDOR_PERF_HINT_END = 0x000011FF,
//reserved for power hints
VENDOR_POWER_HINT_BEGIN = 0x00001200,
VENDOR_POWER_HINT_END = 0x000015FF,
VENDOR_HINT_END = 0x00002000,

3.3.2 Java API

3.3.2.1 Prototype

```
public int perfHint(int hint, String pkg_name, int duration, int reserved)
```

3.3.2.2 Parameters

- `hint`- ID identifying each action uniquely, like launch and scroll.
- `pkg`- Package name of the app, who triggered the hint
- `duration`- Duration of the hint perflock.
- `Reserved(Type)`- To differentiate between similar action, like horizontal scroll vs vertical scroll

3.3.2.3 Return

This function returns 0 on success and -1 on failure.

3.3.3 Native API

3.3.3.1 Prototype

```
int perf_hint(int hint, char *pkg, int duration, int type)
```

3.3.3.2 Parameters

- hint- ID identifying each actionlike launch, scroll and so on.
- Pkg- Package name of the app, who triggered the hint
- Duration- Duration of the hint perflock.
- Type- To differentiate between similar action, like horizontal scroll/vertical scroll

3.3.3.3 Return

This function returns 0 on success and -1 on failure.

3.3.4 Reference

NOTE: Define new hint within this BEGIIN/END limit in their respective category.

3.4 Examples

3.4.1 Java API example

1. Add import android.util.BoostFramework; in your Java source file
2. Create the BoostFramework class object
3. Use `perfLockAcquire` to request the optimizations required
4. Use `perfLockRelease` to release the lock
5. Use `perfHint` for hint-based request.

3.4.1.1 Example 1

Request perflock for minimum of two cores (on significant cluster) and set the minimum frequency for large cluster to 1.5 GHz.

```
BoostFramework mPerf = new BoostFramework();
int duration = 3000;
int list[];

list[0] = MPCTLV3_MIN_ONLINE_CPU_CLUSTER_BIG;
list[1] = 2;

list[2] = MPCTLV3_MIN_FREQ_CLUSTER_BIG_CORE_0
list[3] = 1500;

mPerf.perfLockAcquire(duration, list);
// Critical section requiring PerfLock
```

NOTE: perflockRelease is not required since PerfLock automatically releases after three seconds.

3.4.1.2 Example 2

Request Perflock for minimum of three cores in one section. Set duration for five seconds and release the perflock before the time elapses. Request Perflock for minimum of two cores in another section. Set duration for three seconds and release the perflock before the time elapses.

```
BoostFramework mPerf = new BoostFramework();
BoostFramework sPerf = new BoostFramework();

int list[];

list[0] = MPCTLV3_MIN_ONLINE_CPU_CLUSTER_BIG;
list[1] = 3;

mPerf.perfLockAcquire(5000, list);

//Critical section requiring PerfLock
```

```

mPerf.perfLockRelease();

// other code in between

list[0] = MPCTLV3_MIN_ONLINE_CPU_CLUSTER_BIG;
list[1] = 2;
sPerf.perfLockAcquire(3000, list);

// Critical section requiring PerfLock

sPerf.perfLockRelease();
```

3.4.1.3 Example 3

Request perfHint for vertical scroll.

```
BoostFramework mPerf = new BoostFramework();
mPerf.perfHint(BoostFramework.VENDOR_HINT_SCROLL_BOOST, currentPackage,
mDuration, BoostFramework.Scroll.VERTICAL);
```

3.4.2 Native API Example

1. libdl and libqti-perfd-client_system needs to be added to LOCAL_SHARED_LIBRARIES of the application’s makefile.
2. Include vendor/qcom/proprietary/android-perf/mp-ctl/mp-ctl.h in LOCAL_C_INCLUDES.
3. Include dynamic linking header in source file, that is, #include <dlfcn.h>
4. libqti-perfd-client_system.so library name for dlopen
5. Link the symbols perf_lock_acq,perf_lock_rel and perf_hint using dlsym
6. Declare an integer variable to store the handle returned by perf_lock_acq and perf_hint
7. Use perf_lock_acq to request the optimizations required
8. Use perf_lock_rel to release the lock
9. Use perf_hint to request hint trigger
10. Unload the library using dlclose on cleanup

3.4.2.1 Example

Using Perflock in a native process to bring up two cores from significant cluster.

In makefile:

```
LOCAL_SHARED_LIBRARIES := libqti-perfd-client_system
LOCAL_C_INCLUDES := vendor/qcom/proprietary/android-perf
```

In source file:

```
#include <dlfcn.h>
#include “mp-ctl/mp-ctlh”
static void *qcopt_handle;
static int (*perf_lock_acq)(int handle,
                            int duration,
                            int list[],
                            int numArgs);
static int (*perf_lock_rel)(int handle);
static int (*perf_hint)(int, char *, int, int);
static int perf_lock_handle;
char opt_lib_path[PATH_MAX] = {0};
//Note: Use "libqti-perfd-client_system.so" for opt_lib_path.

if((qcopt_handle = dlopen(opt_lib_path, RTLD_NOW)) == NULL) {
      error_out();
} else {
      perf_lock_acq = (int(*)(int, int, int*,int))dlsym(
      qcopt_handle, "perf_lock_acq");
      perf_lock_rel = (int(*)(int))dlsym(
      qcopt_handle, "perf_lock_rel");
      perf_hint = dlsym(qcopt_handle, "perf_hint");
}

int perf_lock_opts[2] =
      {MPCTLV3_MIN_ONLINE_CPU_CLUSTER_BIG ,0x2};

perf_lock_handle = perf_lock_acq(perf_lock_handle,
                                 0, perf_lock_opts, 2);

// Critical section requiring PerfLock
perf_lock_rel(perf_lock_handle);
```

A References

A.1 Acronyms and terms

Acronym or term	Definition
API	Application programming interface